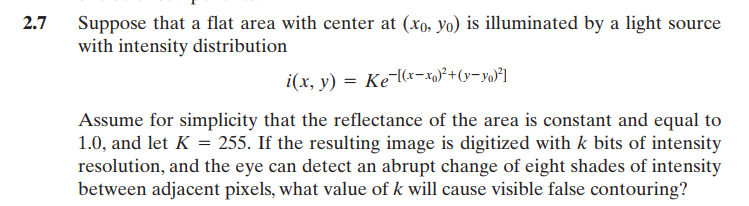
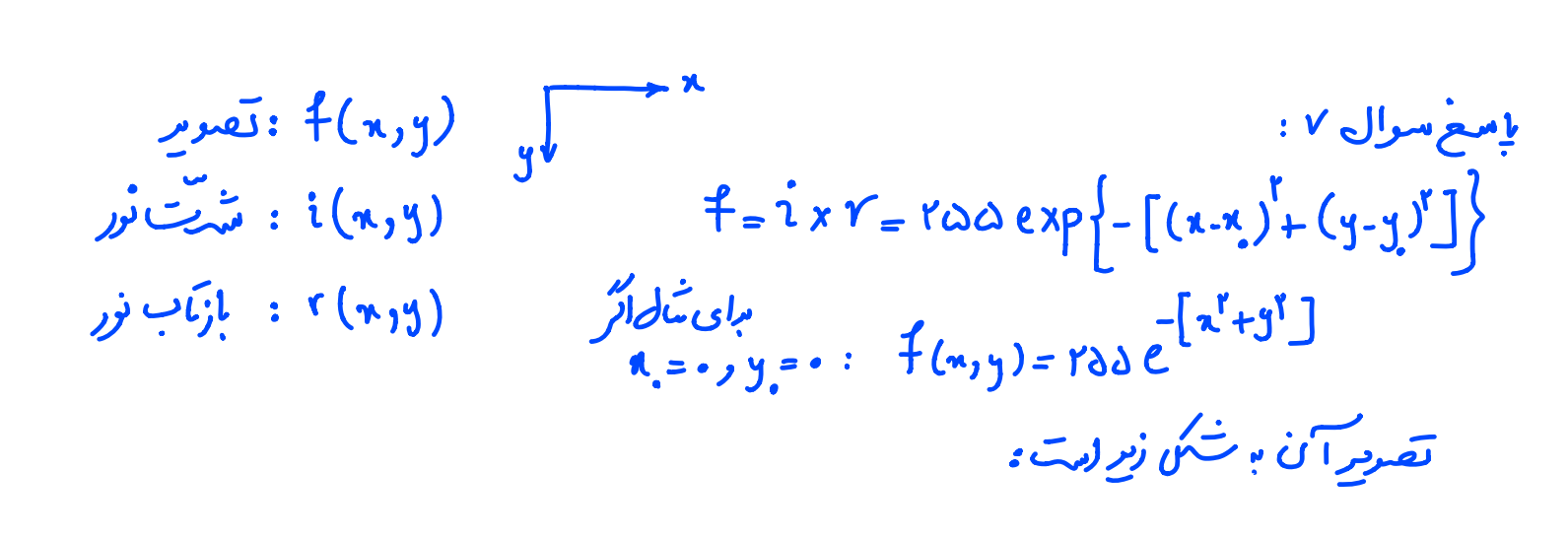
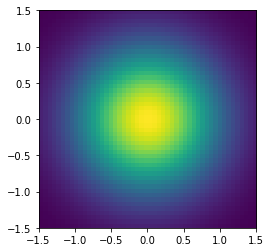
# سوال 1

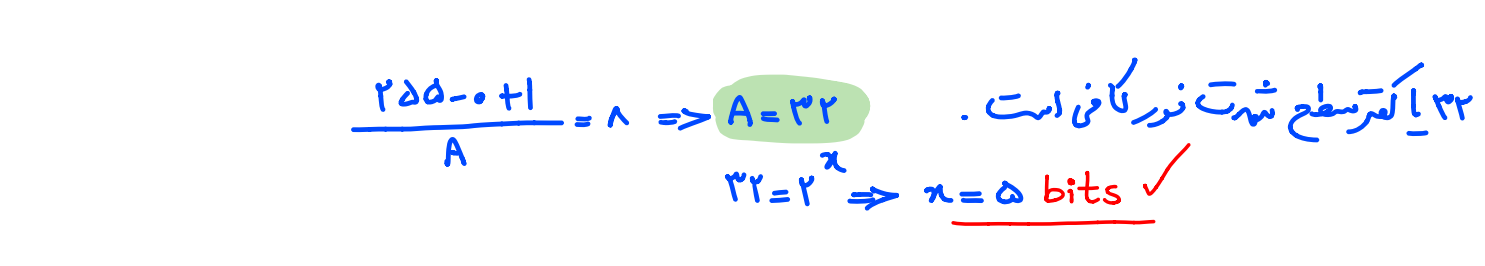


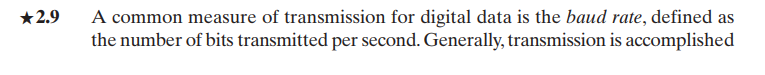


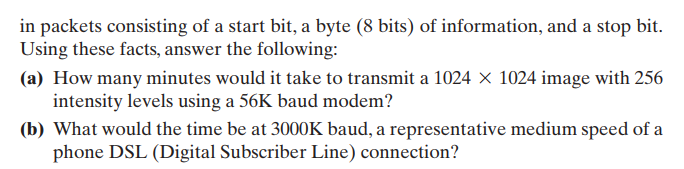
جواب 7:



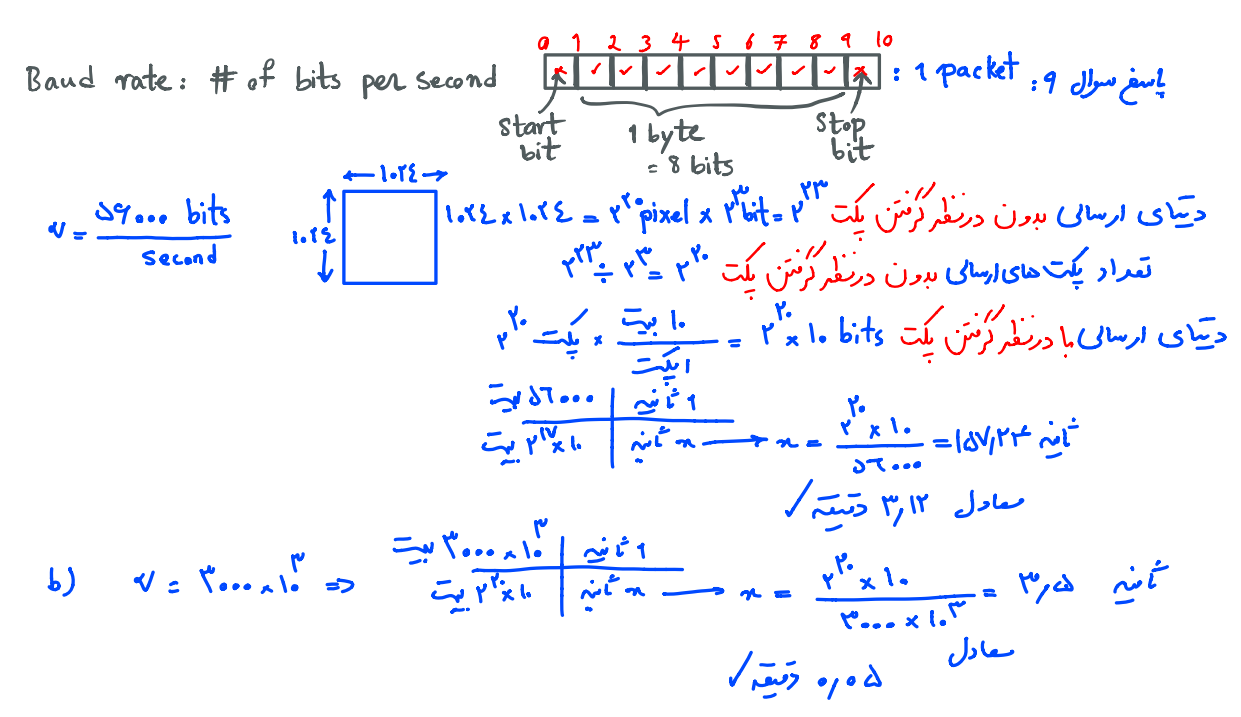




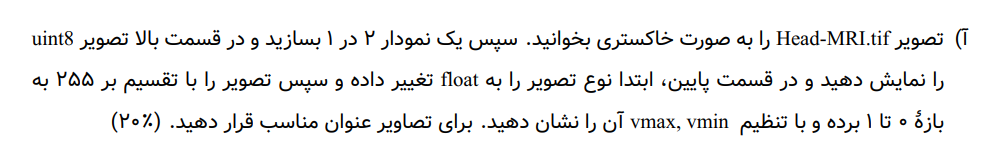




جواب 9:

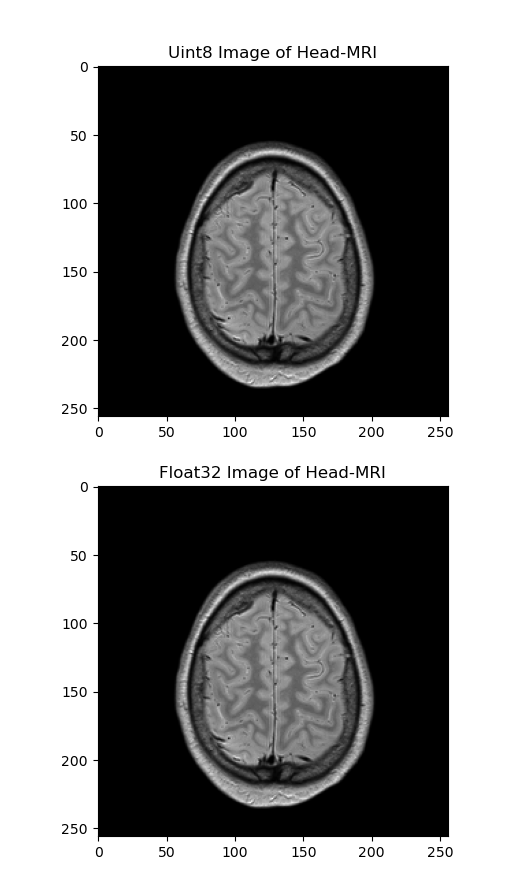


# سوال 2

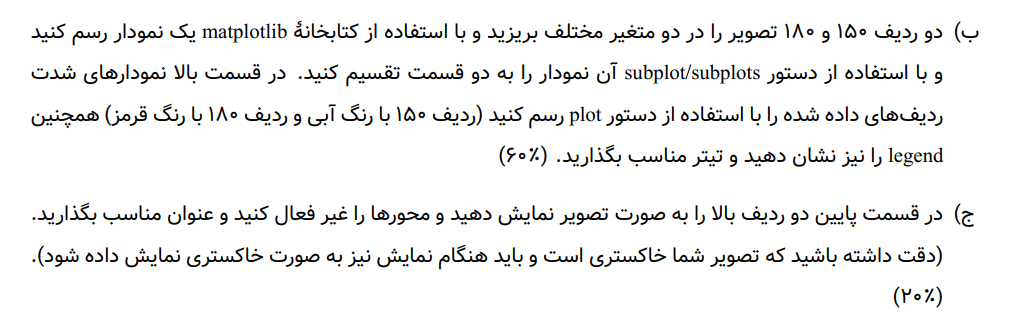


جواب: ابتدا تصویر را بصورت خاکستری میخوانم. از آن جایی که اطلاعات هر پیکسل از تصویر بصورت uint8 و از کلاس نامپای می‌باشد آن ها را با استفاده از تابعی موجود در کتابخانه نامپای بصورت float32 ذخیره میکنم. در ادامه بر 255 تقسیم میکنم و آن را نمایش می‌دهم.

خروجی برنامه:



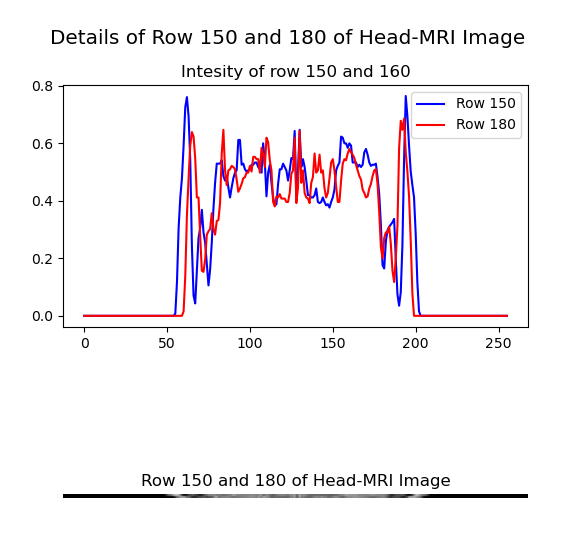
تصویر 1



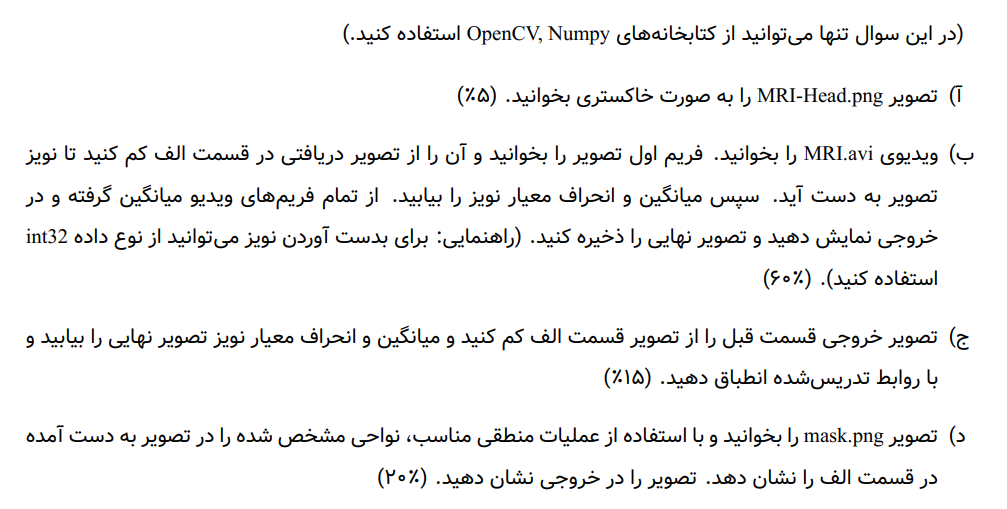
جواب: برای نوشتن این برنامه باید ابتدا سطر های 150 و 180 را از تصویر استخراج کنیم. برای این کار باید به این نکته توجه کنیم که شمارش از صفر شروع میشود و سطر 150 شماره 149 را دارد. این سطر ها را در دو متغیر متفاوت ذخیره کردم و به سادگی آن ها را رسم کردم.

برای بخش (ج) باید ردیف های انتخاب شده را نشان دهیم که با رسم شدت تفاوت دارد. برای به تصویر کشیدن این ردیف ها یک بعد به ارایه اضافه کردم تا آرایه قابل به تصویر کشیدن باشد. سپس برای این که با دستور کار منطبق باشم، دو سطر انتخابی را با هم stack کردم تا بتوانم آن ها را با یکدیگر در یک subplot نمایش دهم.

خروجی:



# سوال 3



جواب: الف و ب و ج) برای این برنامه ابتدا تصویر را میخوانم. سپس ویدیو را بارگیری کرده و فریم اول را از ویدیو میخوانیم. این فریم 3 کانال دارد و آن را به یک کانال یعنی Grayscale تبدیل کردم. سپس تصویر لود شده را از فریمی که ذخیره کرده ام، کم کردم و به آن نویز میگوییم. میانگین و انحراف معیار نویز را با توابع کتابخانه نامپای می‌توان حساب کرد. در نهایت فریم اول، تصویر اصلی و نویز حساب شده را نمایش میدهم. با مشاهده خروجی میتوان مطالبی که سرکلاس تدریش شد را به وضوح مشاهده کرد: میانگین و انحراف معیار نویز در حالت ثانویه ( زمانی که تصاویر ویدیو میانگین گیری شده اند) بسیار کمتر از حالت اولیه ( زمانی که فقط فریم اول ویدیو انتخاب شده است) می‌باشد.

د) در این قسمت ابتدا تصویر ماسک را لود میکنیم و سپس آن را روی تصویر اصلی بصورت فرآیند شرطی AND اعمال میکنیم و در انتها نمایش می‌دهیم.

خروجی:

در صفحه بعد مشاهده کنید.

\_\_\_\_\_\_\_\_ Noise \_\_\_\_\_\_\_

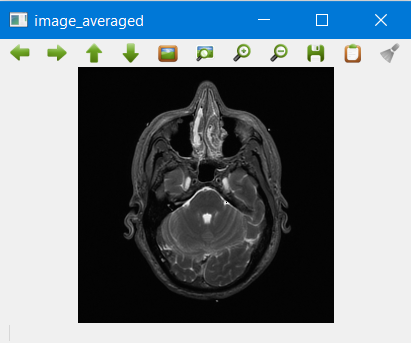
min:-52 max:38 shape:(256, 256)

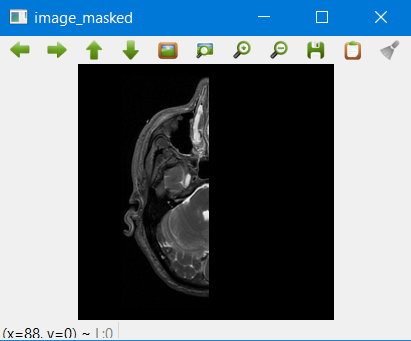
mean:0.8322601318359375 std:8.113755210440392 type:<class 'numpy.int16'>

\_\_\_\_\_\_\_\_ noise2 \_\_\_\_\_\_\_

min:-10.85 max:5.066 shape:(256, 256)

mean:0.73191845703125 std:2.5521150444558436 type:<class 'numpy.float64'>





# سوال 4



جواب: ابتدا تصویر را بارگیری کردم. سپس تصویر را نمایش دادم و اطلاعات آن را در خروجی چاپ کردم (خروجی1). سپس تصویر خوانده شده را به فضای رنگی خاکستری بردم و ابعاد تصویر و نوع هر داده آن را در خروجی چاپ کردم (خروجی 1).

در بخش بعدی تابع با دو ورودی را تعریف کردم. این تابع تصویر را کوانتیزه می‌کند. با توجه به بیت های ورودی با توجه به رابطه کوانتیزه کردن تصویر را کوانتیزه کردم. بیت های تصویر را تقسیم بر 2 به توان 8 منهای بیت ها کرده و floor گرفتم و ضربدر2 به توان 8 منهای بیت ها کرده و با floor عه 2 به توان 8 منهای بیت ها تقسیم بر 2 جمع کردم (رابطه کوانتیزاسیون). نتیجه عملیات بالا را رنک ترانسفورم کردم تا به ترتیب 0و1و2والخ بشوند. حاصل تمام این مراحل را از تابع بازمیگردانیم. در مرحله آخر پس از رسم شکل ها و اعمال threshold ‌نتایج عملیات ها را به تصویر کشیدم (خروجی 2).

تصویر را در خروجی به دست آورید. )

خروجی 1:

\_\_\_\_\_\_\_\_ Color Image \_\_\_\_\_\_\_

min:0 max:255 shape:(366, 409, 3)

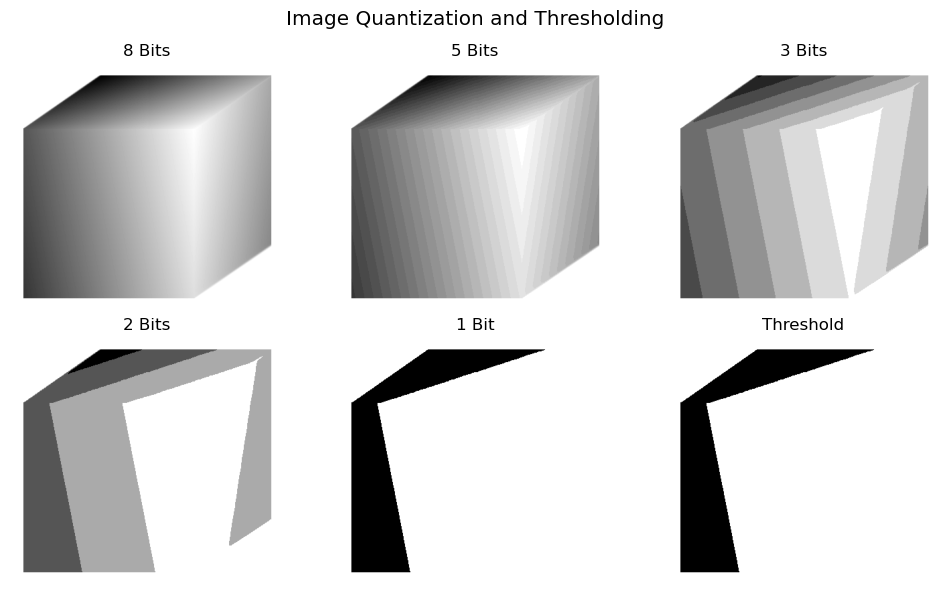
mean:189.1512329596822 std:83.1740414164082 type:<class 'numpy.uint8'>

\_\_\_\_\_\_\_\_ Image Grayscale \_\_\_\_\_\_\_

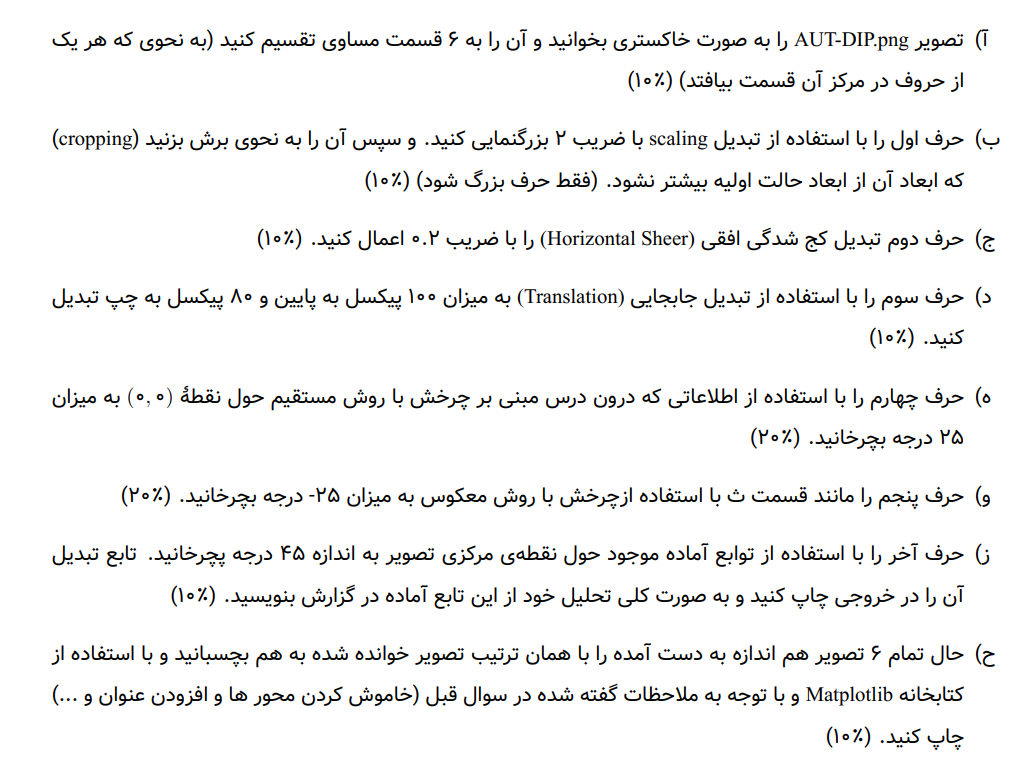
min:29 max:255 shape:(366, 409)

mean:189.84672064344596 std:53.9358216339193 type:<class 'numpy.uint8'>

خروجی 2:



# سوال 5



تصویر را به صورت خاکستری خواندم و آن را به شش قسمت مساوی تقسیم کردم به این صورت که ابتدا طول و عرض تصویر را بدست آورم و با تقسیم کردم طول بر 3 و عرض بر 2 و استفاده از selection آرایه تصویر آن را به شش تصویر تقسیم کردم بطوری که اکنون هر یک از حروف در مرکز آن قسمت افتاده است.

حرف اول را با استفاده از تابع resize یا تبدیل scaling به اندازه 2 برابر بزرگنمایی کردم. ضابطه تبدیل بزرگنمایی به این صورت است:

و سپس آن را با دانستن طول و عرض تصویر جدید و انتخاب کردن مرکز آن بصورت آرایه ای برش زدم و ابعاد برش را طوری باید تنظیم می‌کردم که ابعاد تصویر نهایی برابر ابعاد تصویر اولیه شود.

در مرحله سوم، به حرف دوم تبدیل کج شدگی افقی را با ضریب 0.2 اعمال کردم. تابع تبدیل آن بصورت زیر است و آن را با استفاده از یکی از توابع کتابخانه opnecv به تصویر اعمال کردم:

در مرحله چهارم حرف سوم را با استفاده از تبدیل جابجایی به میزان ۱۰۰پیکسل به پایین و 80 پیکسل به چپ انتقال دادم. تابع تبدیل آن بصورت زیر است و آن را با استفاده از یکی از توابع کتابخانه opnecv به تصویر اعمال کردم:

در مرحله پنجم حرف چهارم را با روش مستقیم به اندازه ۲۵درجه چرخاندم. تابع تبدیل آن بصورت زیر است و آن را با استفاده از یکی از توابع کتابخانه opnecv به تصویر اعمال کردم:

در مرحله ششم به شکل بالا ولی به میزان منفی تصویر را چرخاندم.

در مرحله هفتم حرف ششم را با استفاده از توابع آماده موجود به اندازه ۴۵درجه چرخاندم. تابع تبدیل

آن به صورت زیر است:

[[ 0.70710678 0.70710678 -103.55339059]

[ -0.70710678 0.70710678 250. ]]

تحلیل من این است که این تابع بجای نوشتن ماتریس تبدیل از ابتدا، فقط با گرفتن زاویه چرخش، ماتریس تبدیل را تولید می‌کند و لازم به پیاده سازی تابع تبدیل نیست و همچنین این تابع به تصویر نهایی یک transform نیز اعمال می‌کند تا وسط بودن تصویر محفوظ بماند و تغییر نکند.

در مرحله هشتم (مرحله آخر) تمام تصویر های تولید شده را به صورت معکوس مرحله اول به هم چسباندم و تصویر (ماتریس) خروجی را نمایش و ذخیره سازی کردم و خروجی به شکل زیر است:

