

**دانشگاه صنعتی امیرکبیر**

**دانشکده مهندسی پزشکی**

**گزارش پردازش تصویر**

**تمرین سری اول**

**دانشجو**

**سیدابوالفضل مرتضوی 9833063**

**آبان 1401**

# سوال 1

## بخش 1

بارگذاری شد.

## بخش 2

pic1=dicom.dcmread('file1.dcm')

im\_type=pic1.Modality

im\_bite=pic1.BitsAllocated

im\_loc=pic1[0x0180015].value

im\_bite\_stored=pic1[0x0280101].value

part1\_list=[im\_type,im\_bite,im\_bite\_stored,im\_loc]

بااستفاده از کد بالا اطلاعات خواسته شده به دست آمد و عکس زیر نشان دهنده خروجی آن است.



## بخش 3

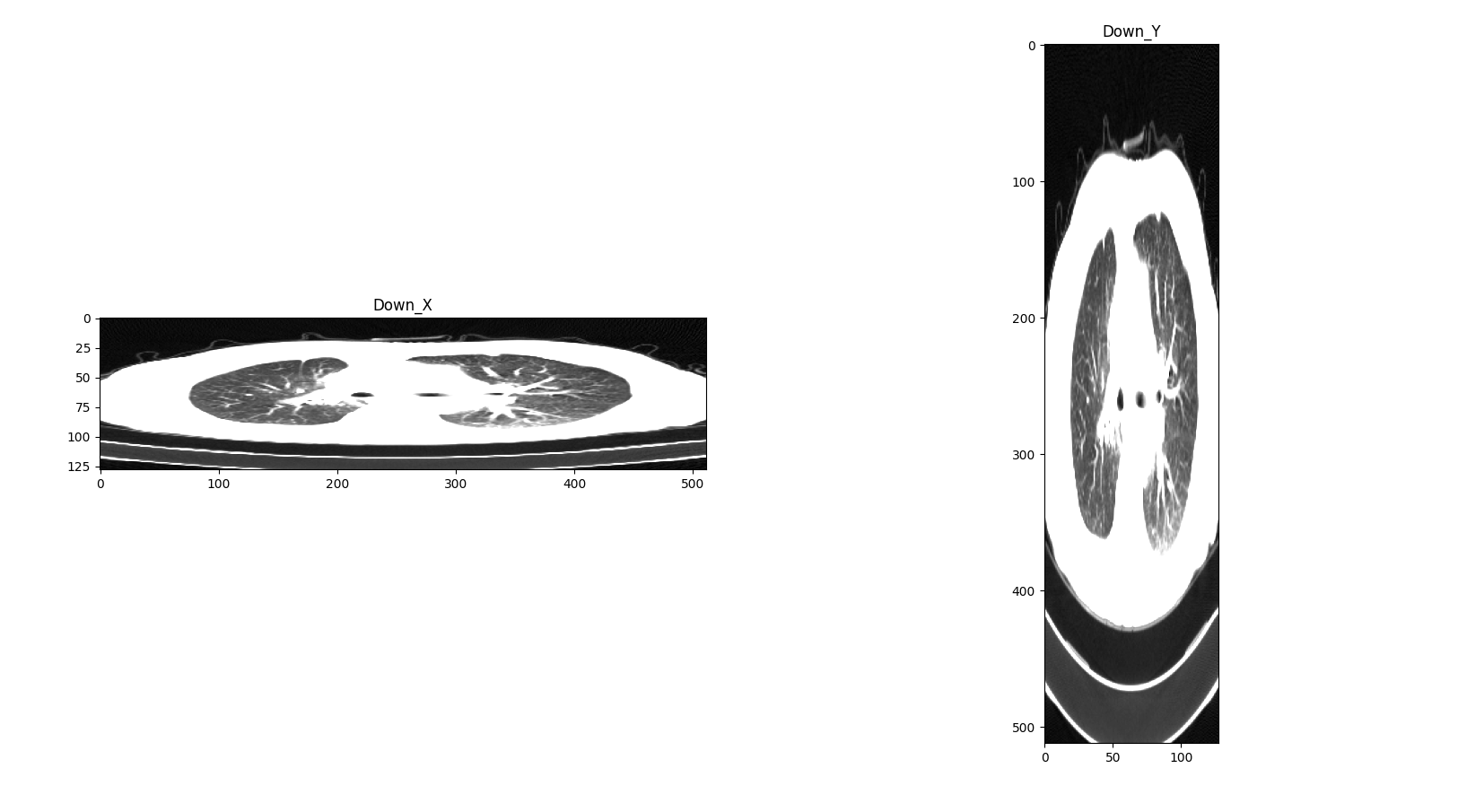
data=pic1.pixel\_array

down\_x=np.array(([data[4\*i,:] for i in range(128)]))

transposed=np.transpose(data)

down\_y=np.transpose(np.array(([transposed[4\*i,:] for i in range(128)])))

بااستفاده از قطعه کد بالا این کار انجام شد. نکته قابل توجه در این میئله این بود که وقتی بدون ترنسپوز درراستای y نمونه کاهی میکردیم این کار به درستی انجام نمیشد و تصویر 90 درجه میچرخید. برای رفع این مشکل از ترنسپوز استفاده شد.

با استفاده از کد بالا این کارانجام شده و تصاویر خروجی در شکل زیر نشان داده شده‌اند. با استفاده از اضافه کردن لایه به لایه آرایه تصویر به یک لیست این کار انجام شده است.

## بخش 4

down\_x1=np.array([data[2\*i,:] for i in range(256)])

down\_xy=np.array([down\_x1[:,2\*i] for i in range(256)])

down\_xy=np.array([down\_xy[:,i] for i in range(256)])

بااستفاده از کد بالا این کار انجام شد نتیجه خروجی در تصویر زیر نشان داده شده است. با استفاده از اضافه کردن لایه به لایه آرایه تصویر به یک لیست این کار انجام شده است.



## بخش 5

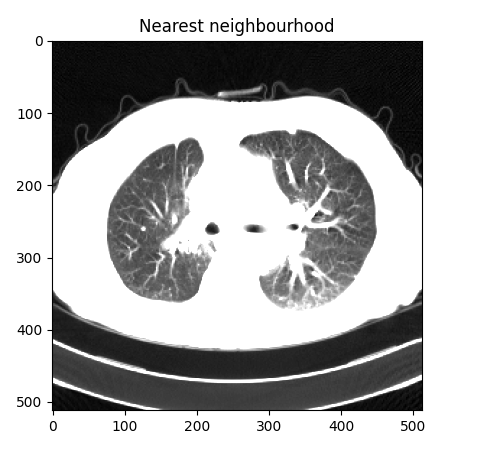
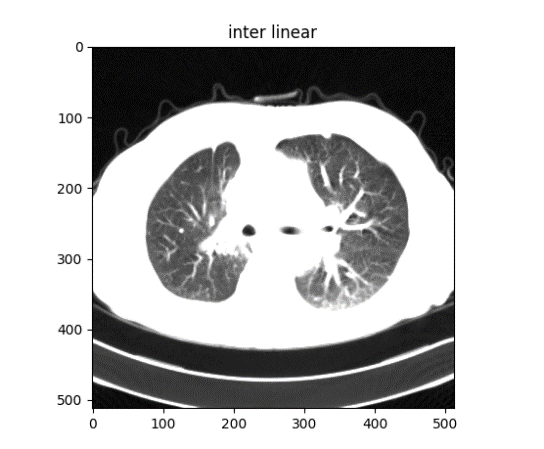
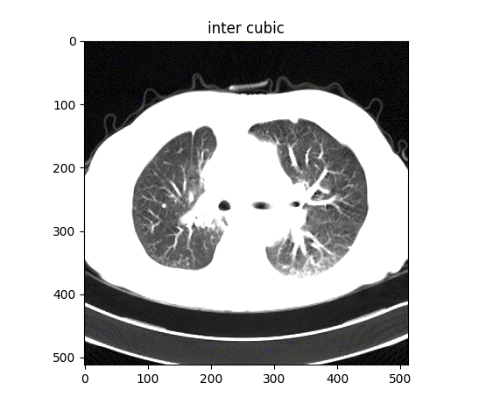
در نمونه کاهی در جهت محور X تصویر به صورت پهن تر درآمده و پهنای اجزای کوچک ، درراستای y کوچک تر شده و عملا حذف شده اند. در راستای محور y هم همین اتفاق افتاده و جزییات در راستای محور x از دست رفته اند. در نمونه کاهی در جهت دومحور تصویر پهن تر یا باریک تر نشده است و فقط کمی از جزئیات از بین رفته اند.

## بخش6

resize\_NN=np.array(Image.fromarray(down\_xy).resize((512,512), Image.NEAREST))

resize\_BL=cv2.resize(down\_xy, (512,512), interpolation=cv2.INTER\_LINEAR)

resize\_BC=cv2.resize(down\_xy,(512,512), interpolation=cv2.INTER\_CUBIC)

 کد های لازم برای انجام این کار در بالا آورده شده است. از متد resize برای این تمرین استفاده شده است.

با توجه به اینکه دربرخی نقاط تصویر تغییر آنی از سیاه به سفید داریم روش نزدیکترین همسایه چندان مناسب نیست. چون می‌تواند باعث کاهش کیفیت تصویر شود. در مورد روش دوخطی این روش در مقایسه با روش قبلی رزولوشن بهتری داردام همچنان کاهش رزولوشن را شاهد هستیم. در روش دو مکعبی رزولوشن تصویر بسیار بهتر از دو روش قبلی است اما سرعت این روش در مقایسه با دو روش قبلی پایین تر است. بسته به نیاز می‌توان از هرکدام از این روش ها استفاده کرد.

# سوال2

## بخش 1

با استفاده از کد زیر این کار انجام شد.

img=[]

cap=cv2.VideoCapture('MRI-Head.avi')

while cap.isOpened():

    i=0

    ret, frame=cap.read()

    if not ret:

        print('end')

        break

    img.append(frame)

img=np.array(img)

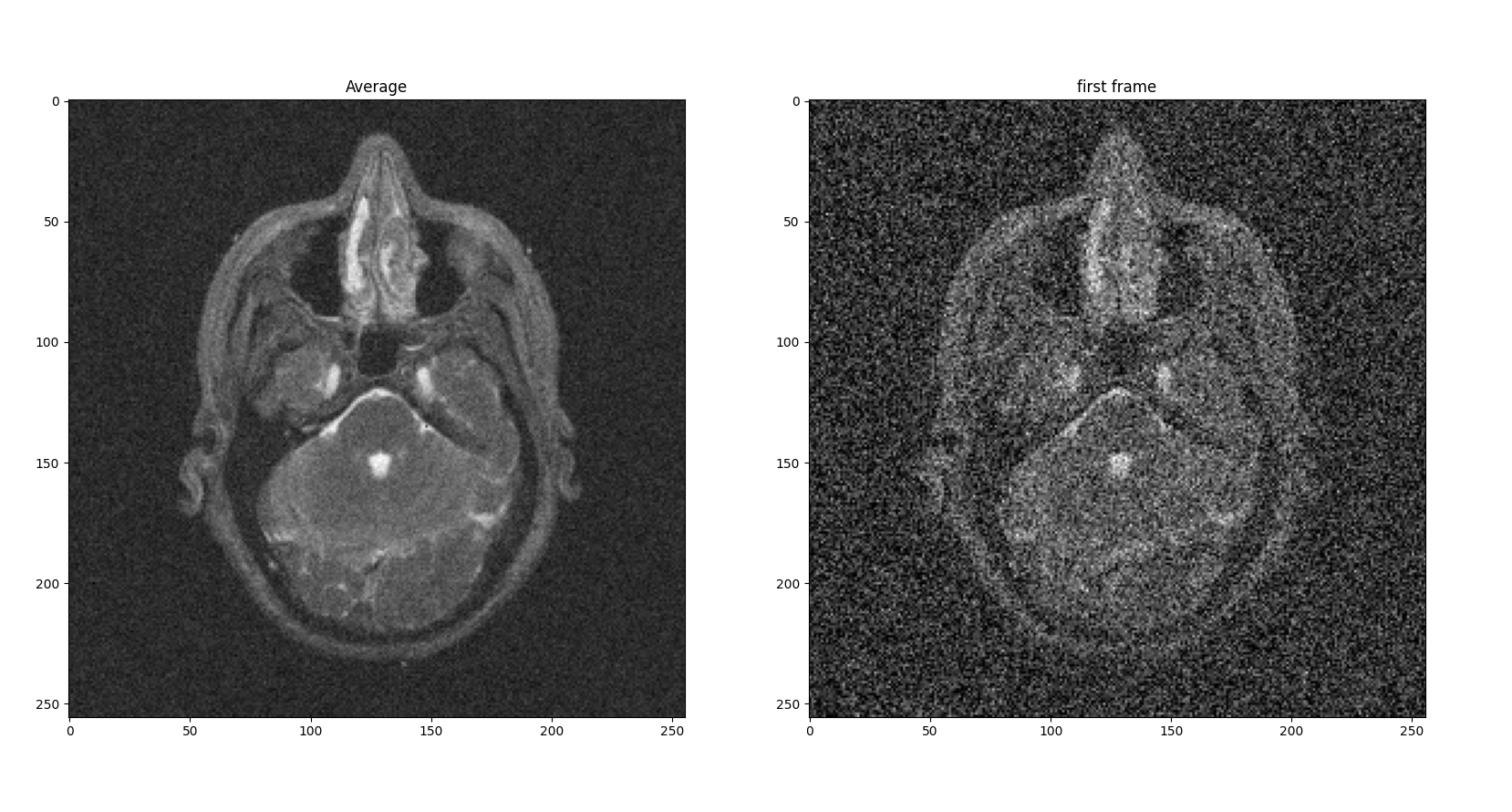
average\_frame=np.zeros(shape=(256,256))

for i in range(256):

    for j in range(256):

        average\_frame[i,j]=np.average(img[:,i,j])

ابتدا ویدیو خوانده شد و سپس فریم به فریم آن در یک لیست ذخیره شد. از تمام ای فریم ها میانگین گرفته شد و نتیجه نهایی در شکل زیر قابل مشاهده است.



## بخش 2

ماسک های گفته شده به ترتیب خوانده شدند و با استفاده از کد زری به تصویر اعمال شدند.

mask1=np.load('mask1.npy')

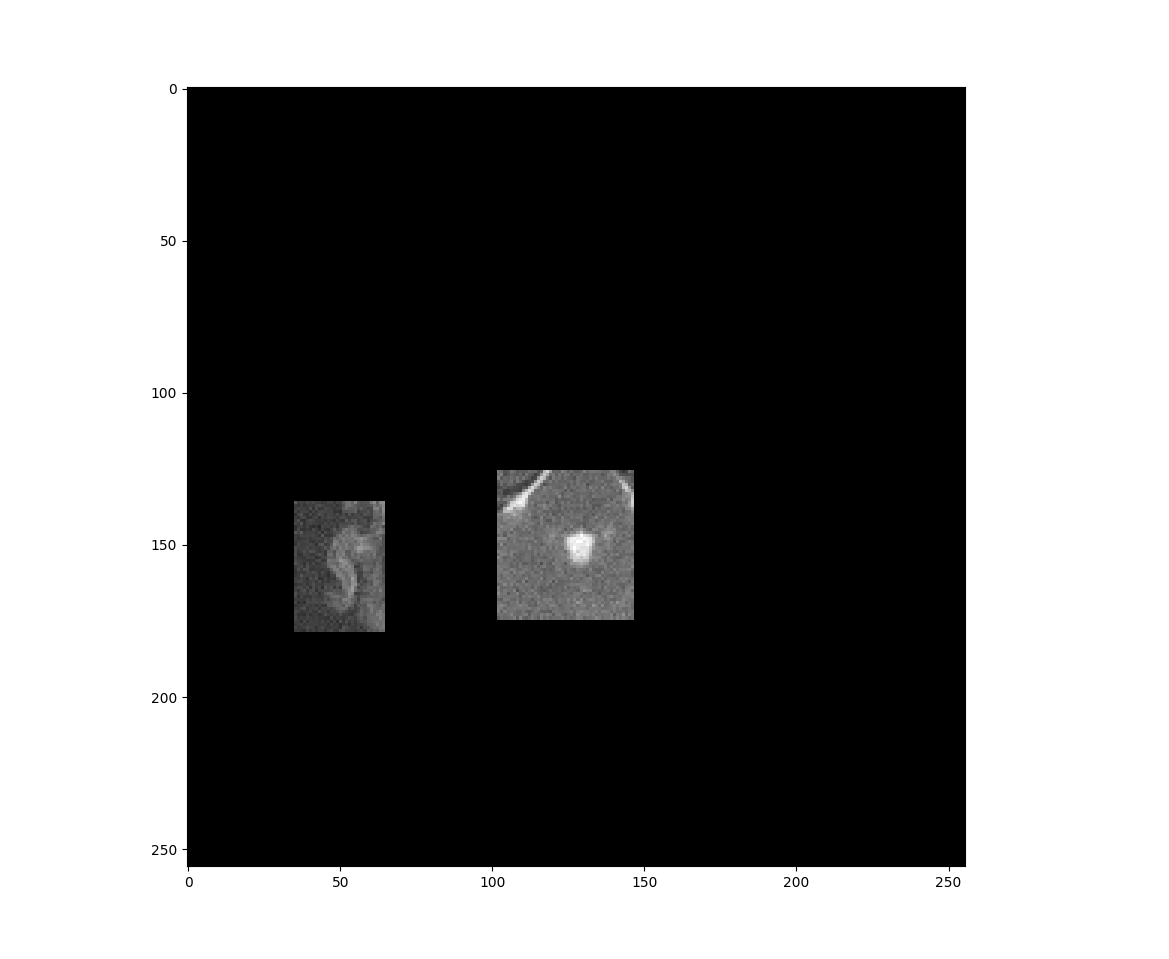
mask2=np.load('mask2.npy')

filtered\_mask1=mask1\*average\_frame

filtered\_mask2=mask2\*average\_frame

all\_filtered=filtered\_mask1+filtered\_mask2

خروجی در شکل صفحه بعد آمده است. این ماسک ها در واقع برای حذف بخش هایی ازتصویر و برجسته کردن یک بخش طراحی شده اند. هرکدام از ماسک ها یک بخش تصویر را برجسته کرده اند و در مجموع دو بخش از تصویر برای نمایش برجسته شده است.



# سوال 3

## بخش 1

ابتدا تصویر در جهت محور y(تئوری درس) نمونه کاهی شده است چرا که طول و عرض آن برابر نیست. سپس اطراف تصویر عمل zero padding انجام شده است در آخر تصویر شیفت یافته است تا به گوشه بالا برسد.

## بخش2

real\_pic=transformed\_pic[0:370,0:256]

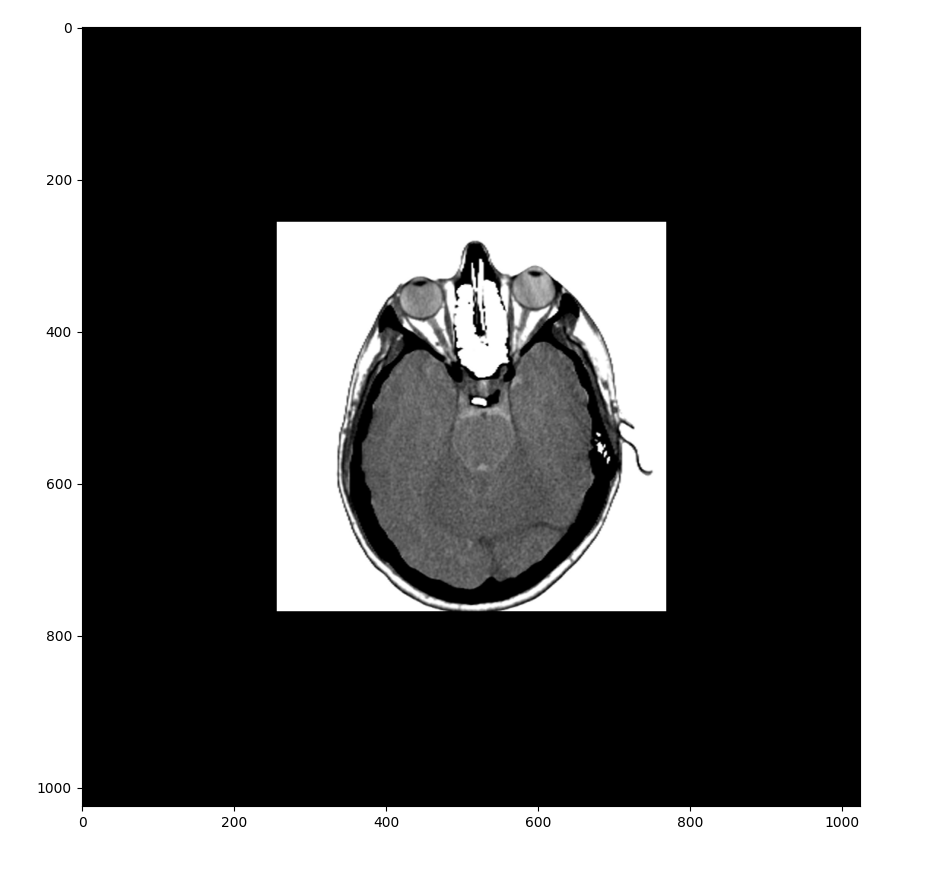
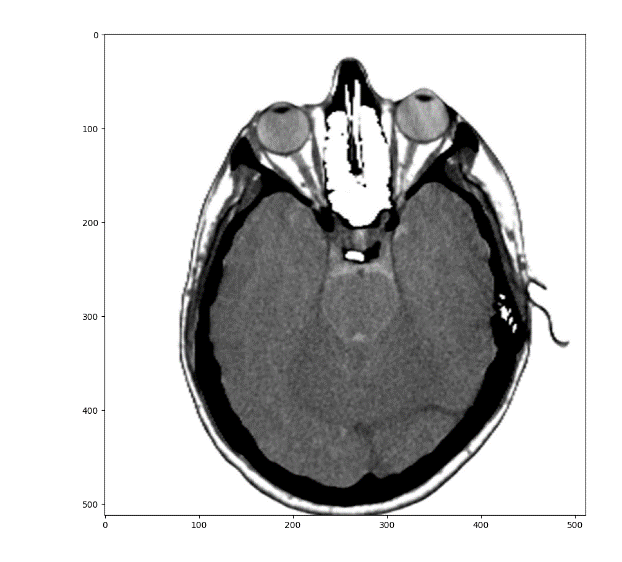
real\_resized\_pic=cv2.resize(real\_pic,(512,512), interpolation=cv2.INTER\_CUBIC)

ابتدا با کد بالا تصویر اصلی مغز را از قسمت های تیره جدا کرده و سپس با روش دو مکعبی تصویر رابه ابعاد 512\*512 برگرداندم. در ادامه با استفاده از کد زیر عمل zero padding انجام شد و به اطراف تصویر 0 افزوده شد تا ابعاد آن به 1024\*1024 برسد.

zeropad\_pic=np.zeros((1024,1024,3)).astype('uint8')

zeropad\_pic[256:768,256:768]=real\_resized\_pic

خروجی مرحله اول و دوم به ترتیب به شکل زیر است.



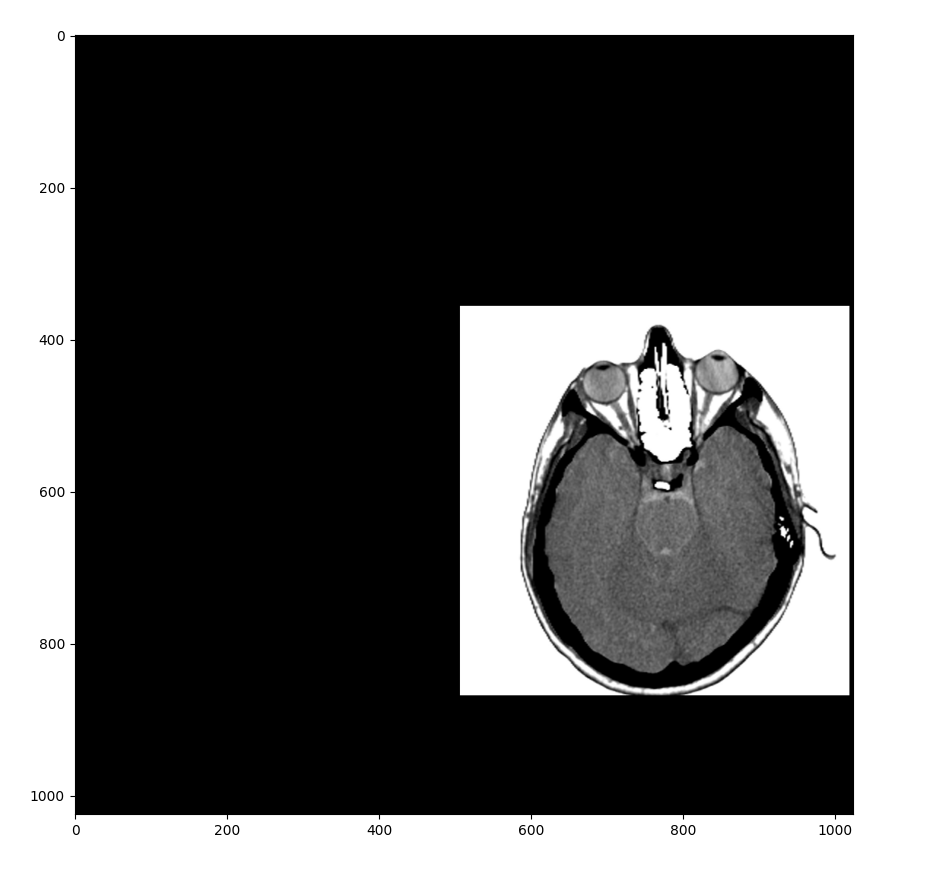
## بخش 3

بااستفاده از کد زیر این عمل انجام شد.

tranlate\_matrix=np.float32(([1,0,250],[0,1,100]))

translated\_pic=cv2.warpAffine(zeropad\_pic,tranlate\_matrix,(1024,1024))

برای انجام این کار از تابع warpAffine که برای اعمال تغییراتی نظیر چرخش و جابه‌جایی و ... نوشته شده است استفاده شد.خروجی این بخش در زیر آمده است.



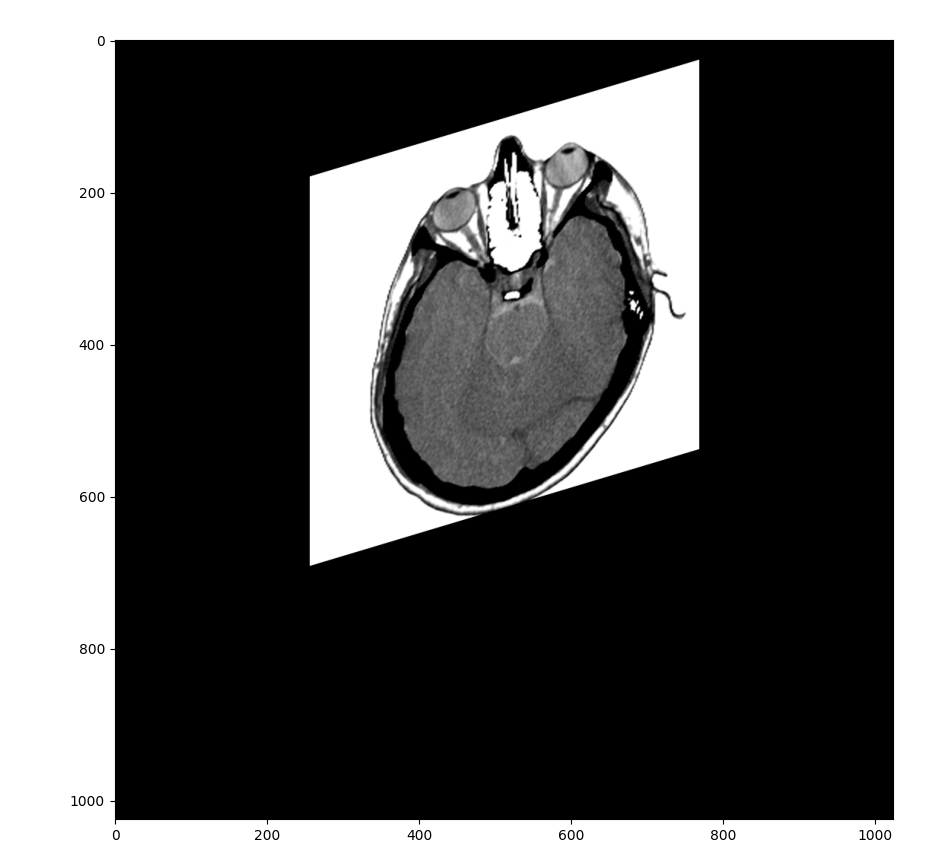
## بخش 4

بااستفاده از کد زیر این عمل انجام شد.

shear\_matrix=np.float32([[1, 0, 0],[-0.3, 1  , 0],[0, 0  , 1]])

sheared\_pic = cv2.warpPerspective(zeropad\_pic,shear\_matrix,(1024,1024))

خروجی قطعه کد بالا به صورت زیر است.



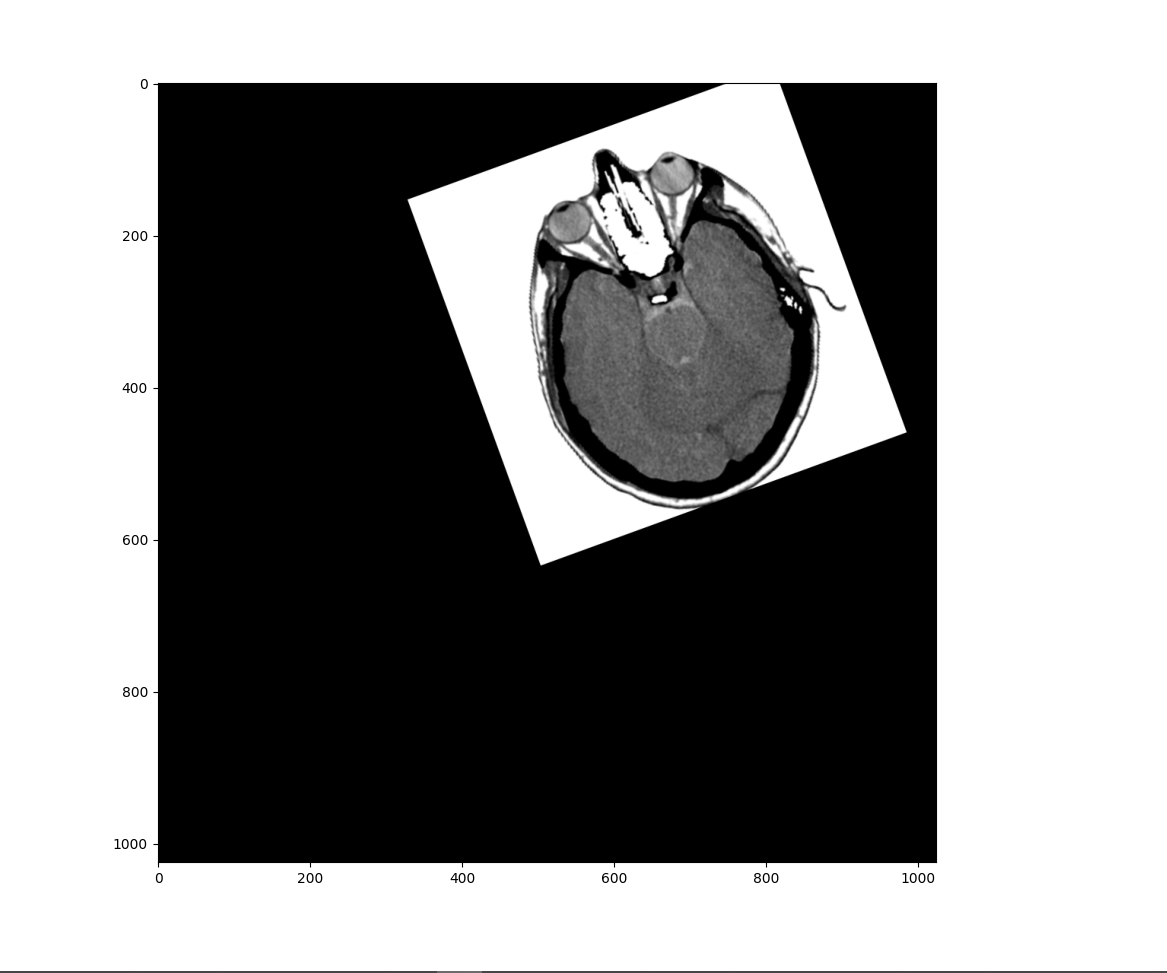
## بخش5

بااستفاده از کد زیر این عمل انجام شد.

rotate\_matrix=cv2.getRotationMatrix2D((0,0), 20, 1.0)

rotated\_pic=cv2.warpAffine(zeropad\_pic, rotate\_matrix, (1024, 1024))

خروجی قطعه کد بالا به صورت زیر است.



# سوال 4

## بخش 1

انجام شد.

## بخش 2

با استفاده از کد زیر این بخش انجام شد.

background=cv2.imread('background-LE.bmp')

fullscale=cv2.imread('fullscale-LE.bmp')

object\_pic=cv2.imread('object-LE.bmp')

avg\_back=[]

avg\_full=[]

for i in range(320):

    avg\_back.append(np.average(background[i,:,:]))

    avg\_full.append(np.average(fullscale[i,:,:]))

avg\_back=np.array(avg\_back)

avg\_full=np.array(avg\_full)

برای میانگین هر تصویر یک لیست تشکیل شد. میانگین هرسطر ازاین لیست در یک عنصر این لیست ذخیره شد.

## بخش 3

بااستفاده از کد زیر این بخش انجام شد.

normalize\_pic=np.zeros(([320,413]))

for i in range(320):

    normalize\_pic[i,:]=(object\_pic[i,:]-avg\_back[i])\*(255/(avg\_full[i]-avg\_back[i]))

برای هرسطر حد پایین و بالا از لیست تعریف شده در بخش قبل به دست آمد.

## بخش 4

تصویر خروجی به صورت زیر است.

