باسمه تعالی

تکلیف سری اول درس سیستم های چندرسانه ای

سارا برادران (شماره دانشجویی: ۹۶۲۴۱۹۳)

بلوک [1] فایل ipynb : کتابخانه ها

در این قسمت کتابخانه های به کار رفته در کد import شده است. به طور کلی از ۳ کتابخانه cv2، numpy و matplotlib استفاده نموده ایم که نصب هر یک از این کتابخانه ها به کمک دستورات زیر قابل انجام است. کتابخانه matplotlib برای نمایش تصاویر، کتابخانه cv2 برای اعمالی از جمله خواندن تصاویر و کتابخانه numpy برای انجام برخی عملیات ها برروی تصاویر مورد استفاده قرار گرفته است که در ادامه به تفصیل به آن ها می پردازیم.

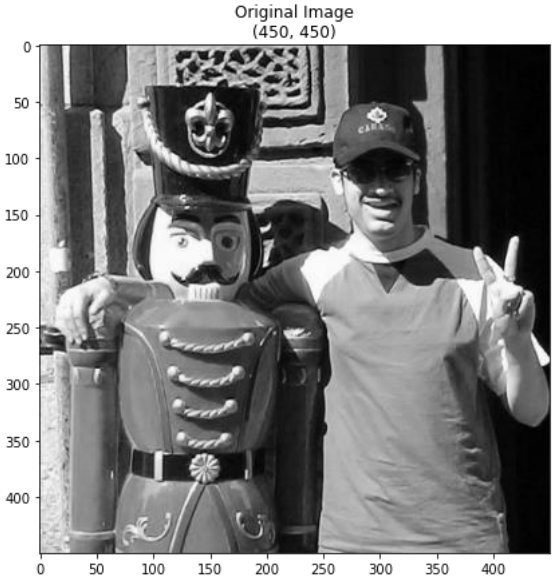
```
pip install numpy

pip install matplotlib

pip install opencv-python
```

بلوک [2] فایل ipynb : تابع Show_Images()

در ابتدا یک تابع تحت عنوان show_images برای نمایش تصاویر به صورت تکی و چندتایی ایجاد شده است. برای نمایش تصویر و پیاده سازی این تابع از کتابخانه matplotlib و دستور imshow استفاده کرده‌ایم. همچنین این تابع به عنوان آرگومان ورودی لیستی از تصاویر، برچسب هر تصویر، و سایز مورد نیاز برای نمایش تصاویر را دریافت می نماید. به علاوه این تابع ابعاد تصاویر دریافتی را در کنار برچسب نام هر تصویر نمایش می دهد. از تابع پیاده سازی شده در مراحل بعدی و برای نمایش تصویر خروجی حاصل از توابع پیاده سازی شده استفاده می کنیم.

بلوک [3] فایل ipynb : خواندن تصویر اصلی و نمایش آن

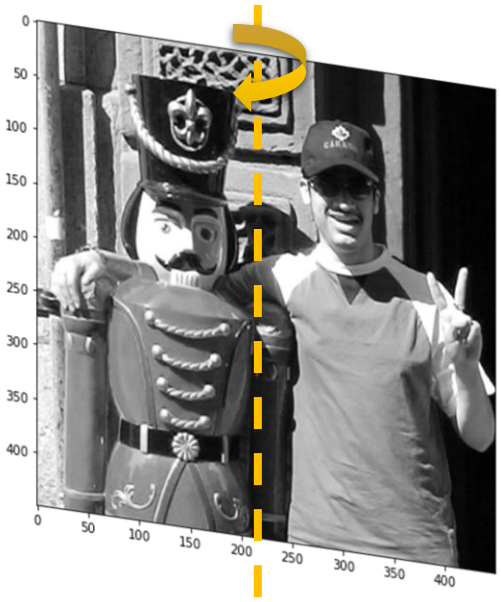
در این قسمت ابتدا به وسیله تابع imread کتابخانه cv2 تصویر Hi.tif را خوانده و درون src_img ذخیره می نماییم سپس به وسیله تابع Show_Images نوشته شده قسمت های پیشین، تصویر اصلی را نمایش داده ایم. تصویر اوریجینال مورد نظر دارای ابعاد ۴۵۰ * ۴۵۰ بوده و مطابق شکل (۱) می باشد.



شکل (۱)

بلوک [4] فایل ipynb : تابع Horizontal_Flip()

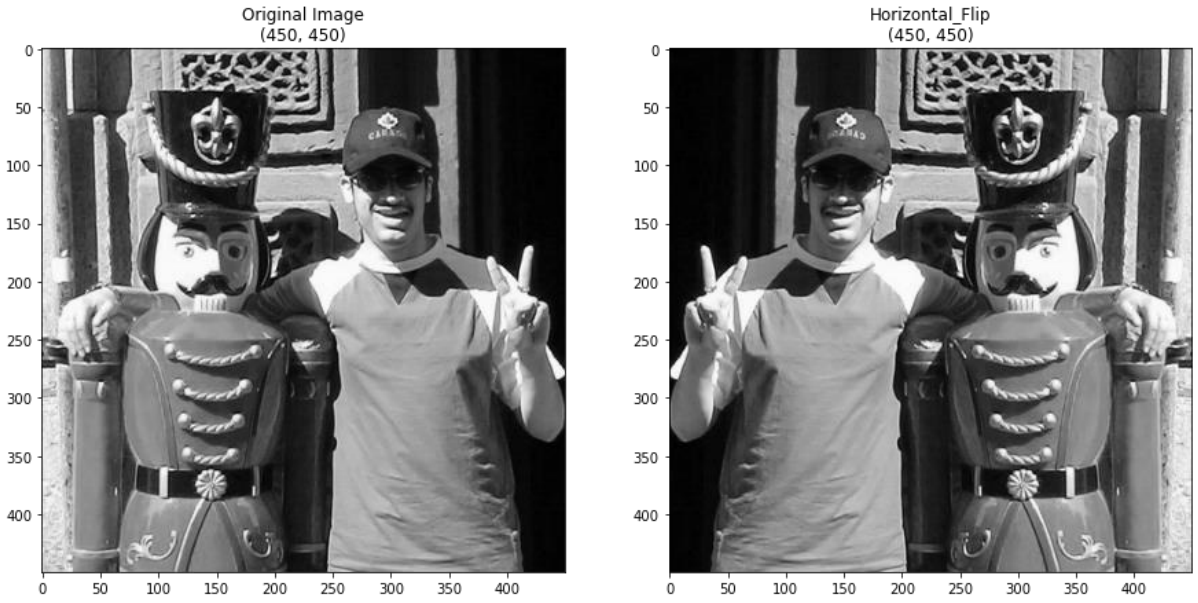
در این قسمت تابعی تحت عنوان Horizontal_flip پیاده سازی شده است که تصویر اصلی را حول محور عمودی دوران می دهد. در این تابع آرایه ای با ابعاد تصویر اولیه ایجاد شده و سپس هر ستون از پیکسل های تصویر اصلی از سمت چپ به راست منتقل می شوند به عبارت دیگر ستون با اندیس ۰ تصویر اصلی در ستون ۴۴۹ تصویر ثانویه، ستون با اندیس ۱ تصویر اصلی در ستون ۴۴۸ تصویر ثانویه و ... قرار می گیرد. در انتها ماتریس تصویر نهایی به وسیله تابع np.array() به فرمت numpy array تبدیل می شود چرا که تصاویر در این فرمت به عنوان ورودی تابع imshow پذیرفته می شوند.



شکل (۲)

بلوک [5] فایل ipynb : فراخوانی تابع Horizontal_Flip() و نمایش تصویر خروجی

در این قسمت تابع Horizontal_Flip فراخوانی شده و src_img به عنوان تصویر اولیه به این تابع پاس داده می شود سپس تصویر حاصل از خروجی تابع و نیز تصویر اولیه به وسیله تابع Show_Images نمایش داده شده است که قابل مقایسه می باشند. دو تصویر نمایش داده شده مشابه شکل (۳) می باشد.



شکل (۳)

بلوک [6] فایل ipynb : تابع Transpose()

در این قسمت تابعی تحت عنوان Transpose پیاده سازی شده است که ستون های تصویر اصلی را به عنوان سطرهای تصویر ثانویه در نظر می گیرد. در این تابع آرایه ای با ابعادی برعکس تصویر اولیه ایجاد شده و سپس هر ستون از پیکسل های تصویر اصلی یک به یک درون سطر های تصویرثانویه قرار می گیرند. در انتها ماتریس تصویر نهایی به وسیله تابع np.array() به فرمت numpy array تبدیل می شود چرا که تصاویر در این فرمت به عنوان ورودی تابع imshow پذیرفته می شوند.

بلوک [7] فایل ipynb : فراخوانی تابع Transpose() و نمایش تصویر خروجی

در این قسمت تابع Transpose فراخوانی شده و src_img به عنوان تصویر اولیه به این تابع پاس داده می‌شود سپس تصویر حاصل از خروجی تابع و نیز تصویر اولیه به وسیله تابع Show_Images نمایش داده شده است که قابل مقایسه می باشند. دو تصویر نمایش داده شده مشابه شکل (۴) می باشد.

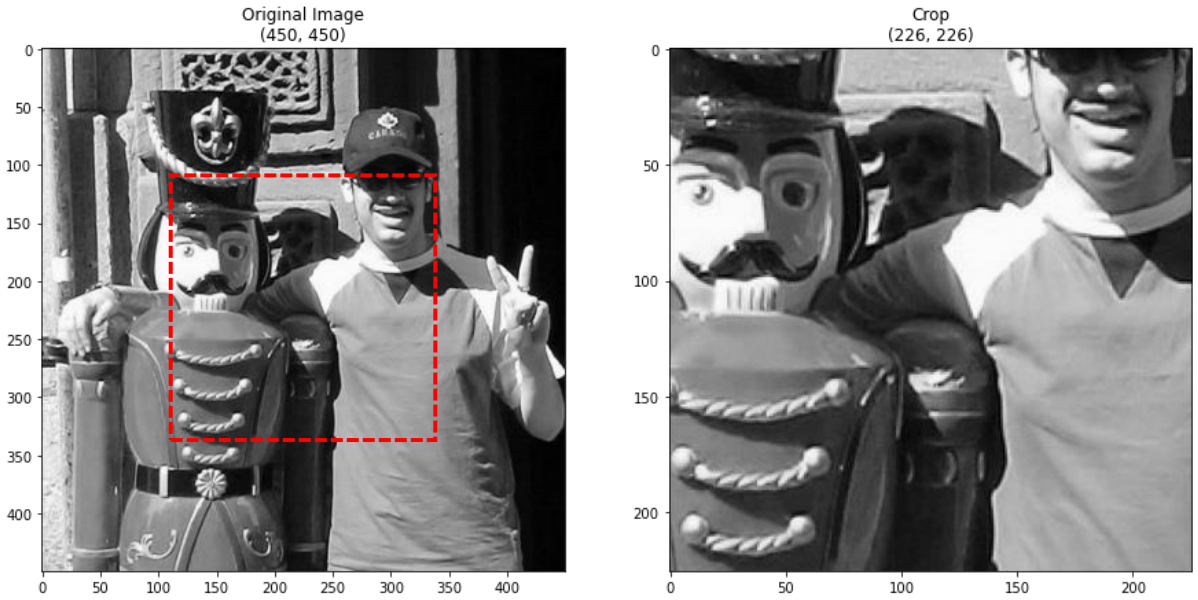
شکل (۴)

بلوک [8] فایل ipynb : تابع Crop()

در این قسمت تابعی تحت عنوان Crop پیاده سازی شده است که تصویر اصلی را به ابعاد دلخواه برش می‌دهد. این تابع علاوه بر تصویر اولیه میزان برش های ۴ طرف تصویر را نیز به عنوان آرگومان ورودی دریافت می‌کند. مثلا اگر بخواهیم ۱۰ درصد از سمت راست تصویر را برش دهیم در اینصورت کافیست آرگومان ورودی r_percent = 0.1 قرار داده شود. آرگومان های r_percent, l_percent, t_percent و b_percent به ترتیب میزان برش از بالا، چپ، راست و پایین تصویر را نشان می دهند. پس از دریافت آرگومان های ورودی پوزیشن قرار گیری ۴ طرف تصویر ثانویه تعیین گشته و به وسیله src_img[top:bottom, left:right] از نقطه top الی bottom بعد اولیه و از نقطه left الی right بعد ثانویه تصویر نخست، جداسازی شده و آرایه خروجی به عنوان تصویر نهایی و crop شده قرار می گیرد.

بلوک [9] فایل ipynb : فراخوانی تابع Crop() و نمایش تصویر خروجی

در این قسمت تابع Crop فراخوانی شده و src_img به عنوان تصویر اولیه و آرگومان های r_percent و l_percent و t_percent و b_percent همگی با مقادیر ۰/۲۵ به این تابع پاس داده می شوند سپس تصویر حاصل از خروجی تابع و نیز تصویر اولیه به وسیله تابع Show_Images نمایش داده شده است که قابل مقایسه می باشند. دو تصویر نمایش داده شده مشابه شکل (۵) می باشد.



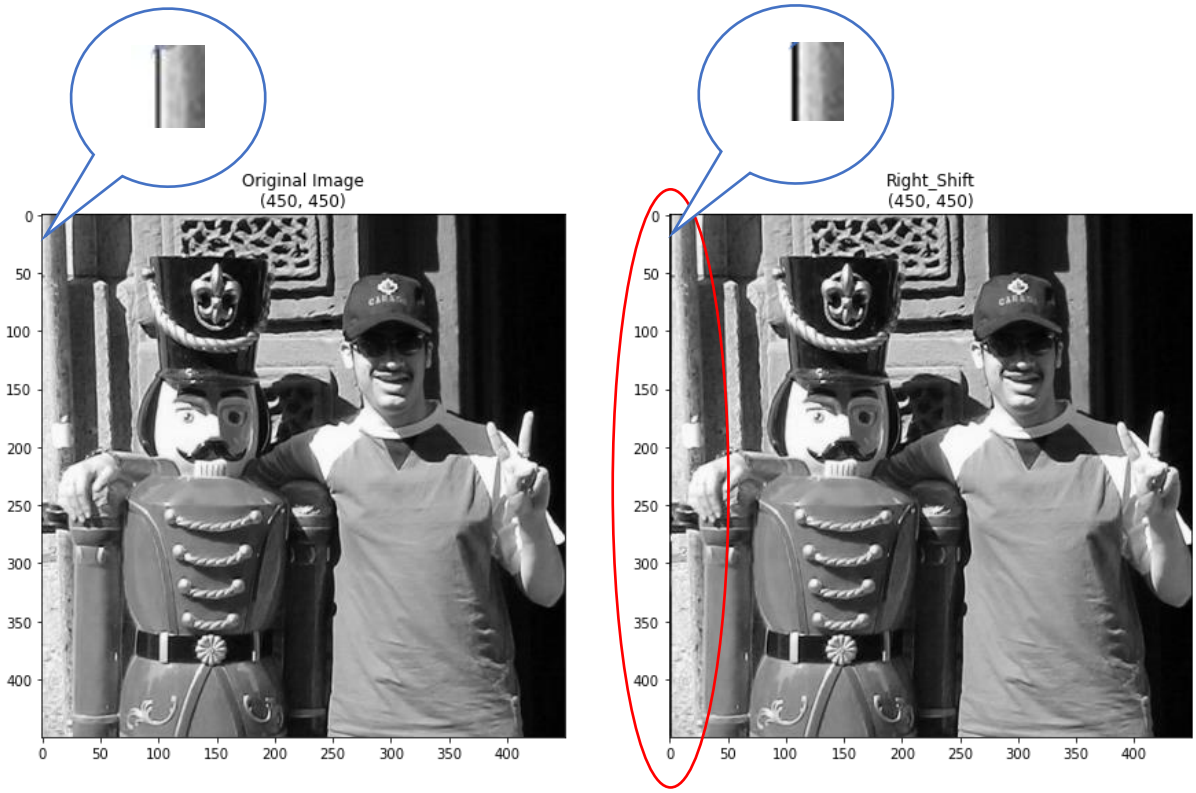
شکل (۵)

بلوک [10] فایل ipynb : تابع ()Right_Shift

در این قسمت تابعی تحت عنوان Right_Shift پیاده سازی شده است که تصویر اصلی را به میزان دلخواه شیفت گردشی به راست می دهد. این تابع علاوه بر تصویر اولیه تعداد پیکسل هایی که قرار است شیفت گردشی به راست داده شود را نیز دریافت می کند. مثلا اگر بخواهیم به اندازه ۱۰ پیکسل تصویر را به سمت راست شیفت گردشی دهیم کافی است آرگومان ورودی pixel=10 قرار داده شود. در این تابع آرایه ای با ابعاد تصویر اولیه ایجاد شده و سپس هر ستون از پیکسل های تصویر اصلی ۲ خانه به سمت راست شیفت داده می شوند به عبارت دیگر new_img[i][j] = src_img[i][(j - pixel) % width] قرار می گیرد. این عملیات منجر می شود پیکسل های ستون ۰ تا ۴۴۷ تصویر اصلی در پیکسل های ۲ تا ۴۴۹ تصویر نهایی قرار گیرند و پیسکل های ستون ۴۴۸ و ۴۴۹ تصویر اولیه به عنوان پیکسل های ستون ۰ و ۱ تصویر نهایی لحاظ شوند. در انتها ماتریس تصویر نهایی به وسیله تابع np.array() به فرمت numpy array تبدیل می شود چرا که تصاویر در این فرمت به عنوان ورودی تابع imshow پذیرفته می شوند.

بلوک [11] فایل ipynb : فراخوانی تابع ()Right_Shift و نمایش تصویر خروجی

در این قسمت تابع Right_Shift فراخوانی شده و src_img به عنوان تصویر اولیه و آرگومان pixel با مقدار ۲ به این تابع پاس داده می شوند سپس تصویر حاصل از خروجی تابع و نیز تصویر اولیه به وسیله تابع Show_Images نمایش داده شده است که قابل مقایسه می باشند. دو تصویر نمایش داده شده مشابه شکل (۶) می باشد.



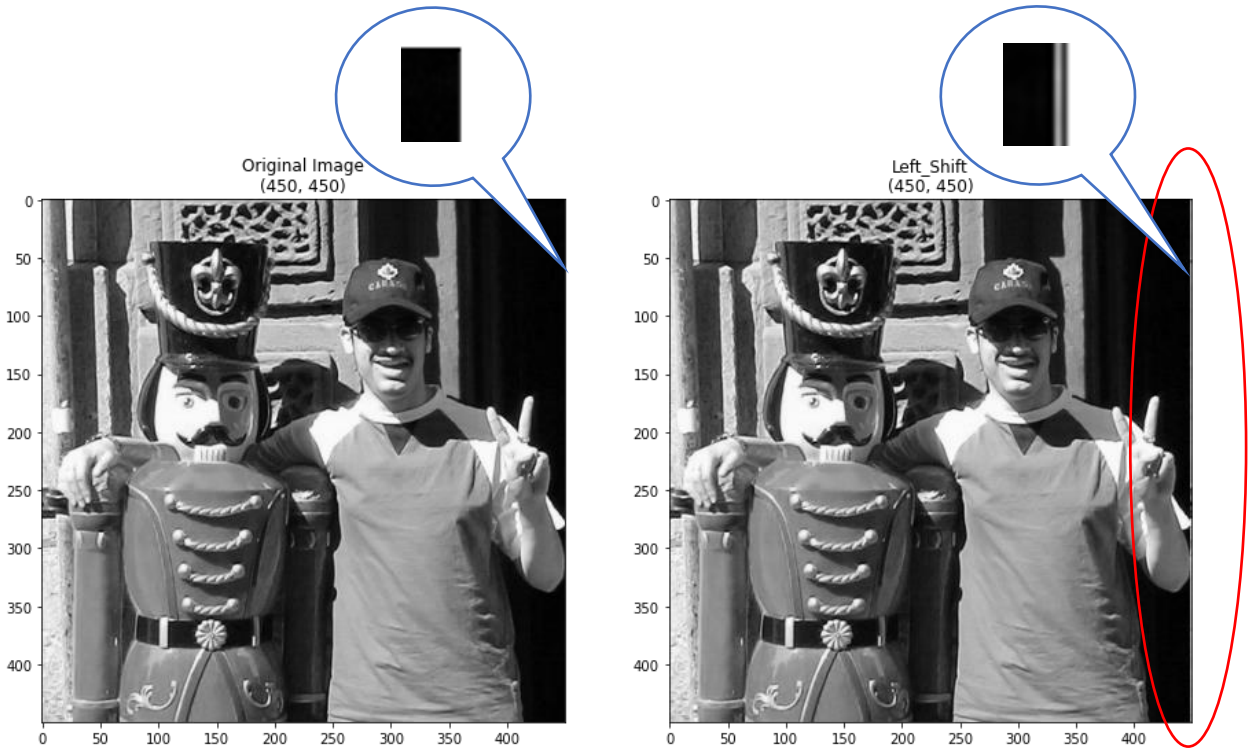
شکل (۶)

بلوک [12] فایل ipynb : تابع Left_Shift()

در این قسمت تابعی تحت عنوان Left_Shift پیاده سازی شده است که تصویر اصلی را به میزان دلخواه شیفت گردشی به چپ می دهد. این تابع علاوه بر تصویر اولیه تعداد پیکسل هایی که قرار است شیفت گردشی به چپ داده شود را نیز دریافت می کند. مثلا اگر بخواهیم به اندازه ۱۰ پیکسل تصویر را به سمت چپ شیفت گردشی دهیم کافی است آرگومان ورودی pixel=10 قرار داده شود. در این تابع آرایه ای با ابعاد تصویر اولیه ایجاد شده و سپس هر ستون از پیکسل های تصویر اصلی ۲ خانه به سمت چپ شیفت داده می شوند به عبارت دیگر new_img[i][j] = src_img[i][(j + pixel) % width] قرار می گیرد. این عملیات منجر می شود پیکسل های ستون ۲ تا ۴۴۹ تصویر اصلی در پیسکل های ۰ تا ۴۴۷ تصویر نهایی قرار گیرند و پیکسل های ستون ۰ و ۱ تصویر اولیه به عنوان پیکسل های ستون ۴۴۸ و ۴۴۹ تصویر نهایی لحاظ شوند. در انتها ماتریس تصویر نهایی به وسیله تابع np.array() به فرمت numpy array تبدیل می شود چرا که تصاویر در این فرمت به عنوان ورودی تابع imshow پذیرفته می شوند.

بلوک [13] فایل ipynb : فراخوانی تابع Left _Shift() و نمایش تصویر خروجی

در این قسمت تابع Left _Shift فراخوانی شده و src_img به عنوان تصویر اولیه و آرگومان pixel با مقدار ۲ به این تابع پاس داده می شوند سپس تصویر حاصل از خروجی تابع و نیز تصویر اولیه به وسیله تابع Show_Images نمایش داده شده است که قابل مقایسه می باشند. دو تصویر نمایش داده شده مشابه شکل (۷) می باشد.



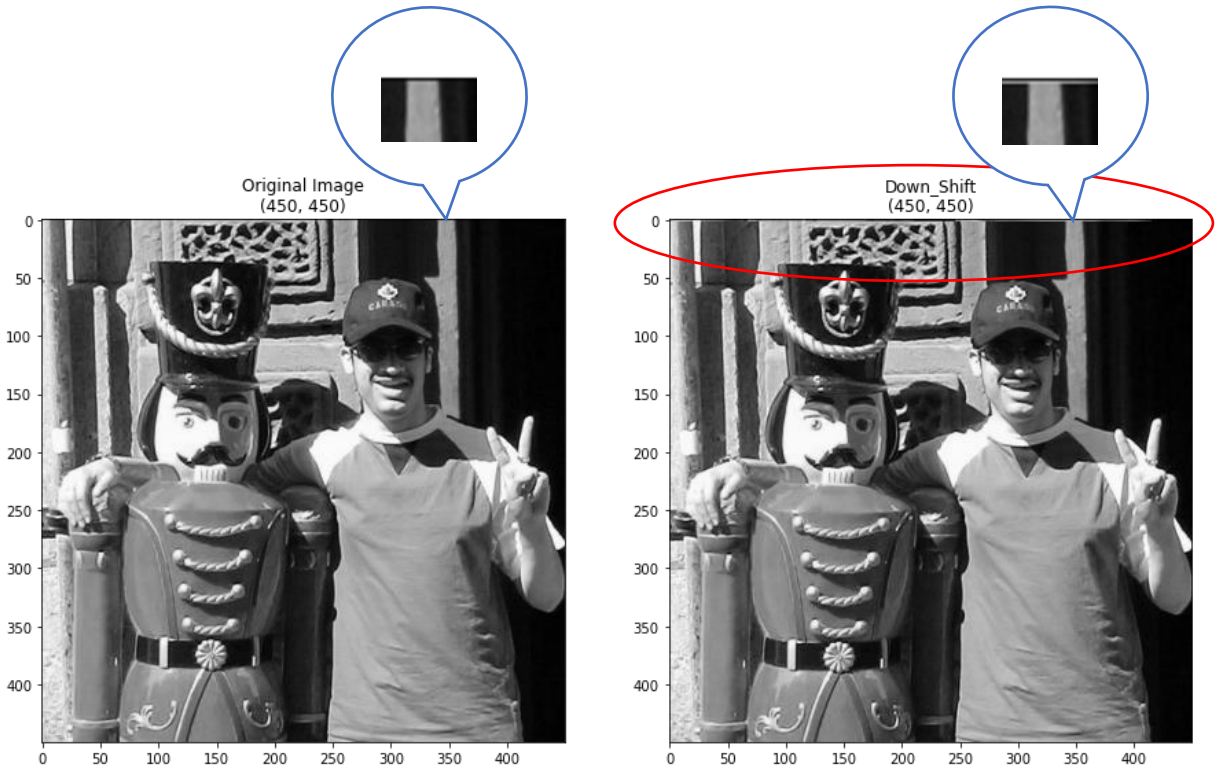
شکل (۷)

بلوک [14] فایل ipynb : تابع Down_Shift()

در این قسمت تابعی تحت عنوان Down_Shift پیاده سازی شده است که تصویر اصلی را به میزان دلخواه شیفت گردشی به پایین می دهد. این تابع علاوه بر تصویر اولیه تعداد پیکسل هایی که قرار است شیفت گردشی به پایین داده شود را نیز دریافت می کند. مثلا اگر بخواهیم به اندازه ۱۰ پیکسل تصویر را به سمت پایین شیفت گردشی دهیم کافی است آرگومان ورودی pixel=10 قرار داده شود. در این تابع آرایه ای با ابعاد تصویر اولیه ایجاد شده و سپس هر سطر از پیکسل های تصویر اصلی ۲ خانه به سمت پایین شیفت داده می شوند به عبارت دیگر new_img[i][j] = src_img[(i – pixel) % height][j] قرار می گیرد. این عملیات منجر می شود پیکسل های سطر ۰ تا ۴۴۷ تصویر اصلی در پیسکل های ۲ تا ۴۴۹ تصویر نهایی قرار گیرند و پیکسل های سطر ۴۴۸ و ۴۴۹ تصویر اولیه به عنوان پیکسل های سطر ۰ و ۱ تصویر نهایی لحاظ شوند. در انتها ماتریس تصویر نهایی به وسیله تابع np.array() به فرمت numpy array تبدیل می شود چرا که تصاویر در این فرمت به عنوان ورودی تابع imshow پذیرفته می شوند.

بلوک [15] فایل ipynb : فراخوانی تابع Down_Shift() و نمایش تصویر خروجی

در این قسمت تابع Down_Shift فراخوانی شده و src_img به عنوان تصویر اولیه و آرگومان pixel با مقدار ۲ به این تابع پاس داده می شوند سپس تصویر حاصل از خروجی تابع و نیز تصویر اولیه به وسیله تابع Show_Images نمایش داده شده است که قابل مقایسه می باشند. دو تصویر نمایش داده شده مشابه شکل (۸) می باشد.



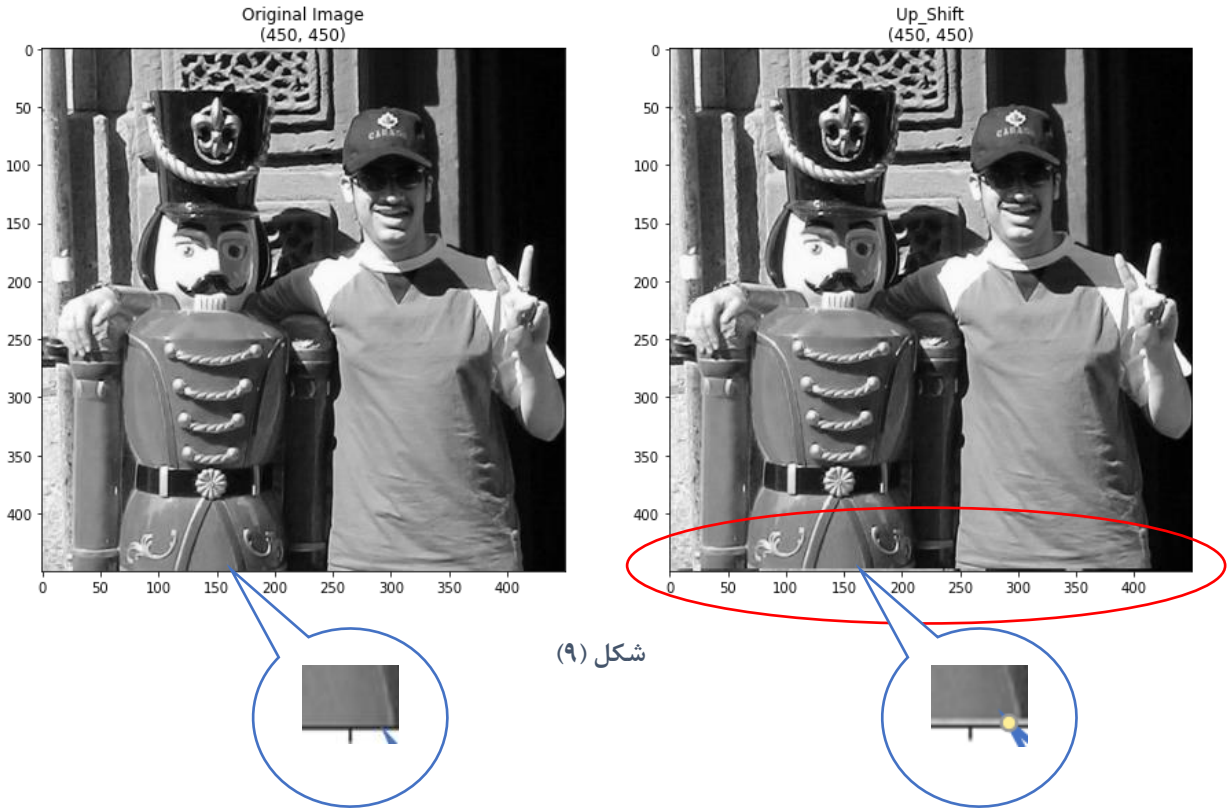
شکل (۸)

بلوک [16] فایل ipynb : تابع ()Up_Shift

در این قسمت تابعی تحت عنوان Up_Shift پیاده سازی شده است که تصویر اصلی را به میزان دلخواه شیفت گردشی به بالا می دهد. این تابع علاوه بر تصویر اولیه تعداد پیکسل هایی که قرار است شیفت گردشی به بالا داده شود را نیز دریافت می کند. مثلا اگر بخواهیم به اندازه ۱۰ پیکسل تصویر را به سمت بالا شیفت گردشی دهیم کافی است آرگومان ورودی pixel=10 قرار داده شود. در این تابع آرایه ای با ابعاد تصویر اولیه ایجاد شده و سپس هر سطر از پیکسل های تصویر اصلی ۲ خانه به سمت بالا شیفت داده می شوند به عبارت دیگر new_img[i][j] = src_img[(i + pixel) % height][j] قرار می گیرد. این عملیات منجر می شود پیکسل های سطر ۲ تا ۴۴۹ تصویر اصلی در پیسکل های ۰ تا ۴۴۷ تصویر نهایی قرار گیرند و پیکسل های سطر ۰ و ۱ تصویر اولیه به عنوان پیکسل های سطر ۴۴۸ و ۴۴۹ تصویر نهایی لحاظ شوند. در انتها ماتریس تصویر نهایی به وسیله تابع np.array() به فرمت numpy array تبدیل می شود چرا که تصاویر در این فرمت به عنوان ورودی تابع imshow پذیرفته می شوند.

بلوک [17] فایل ipynb : فراخوانی تابع ()Up _Shift و نمایش تصویر خروجی

در این قسمت تابع Up _Shift فراخوانی شده و src_img به عنوان تصویر اولیه و آرگومان pixel با مقدار ۲ به این تابع پاس داده می شوند سپس تصویر حاصل از خروجی تابع و نیز تصویر اولیه به وسیله تابع Show_Images نمایش داده شده است که قابل مقایسه می باشند. دو تصویر نمایش داده شده مشابه شکل (۹) می باشد.



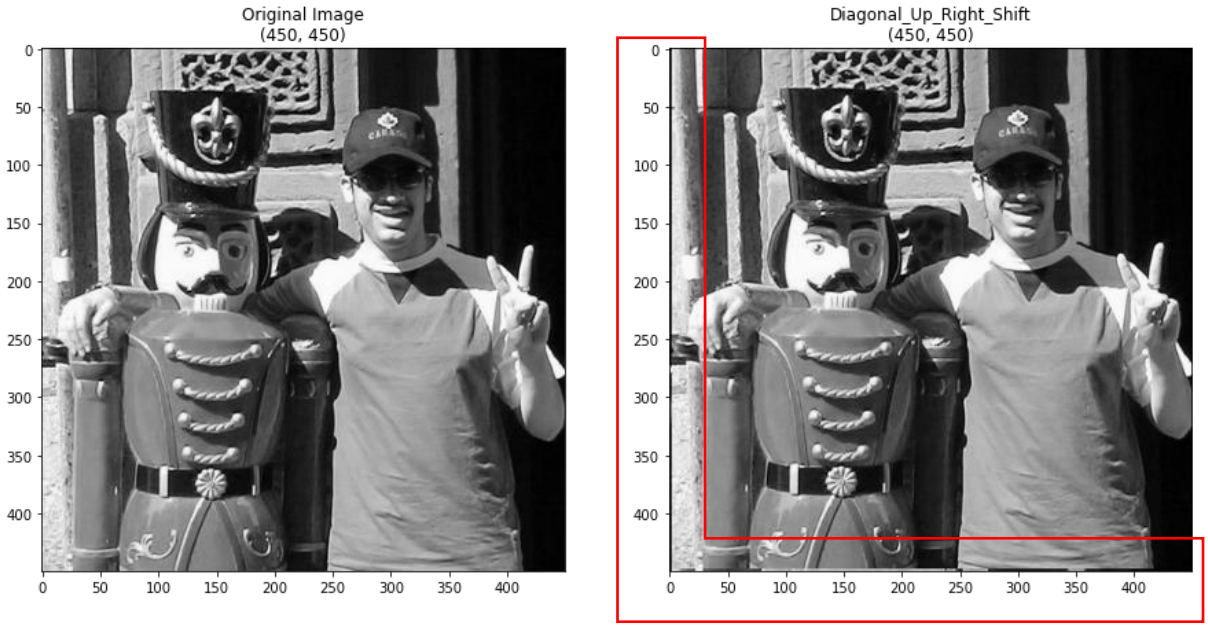
شکل (۹)

بلوک [18] فایل ipynb : تابع Diagonal_Up_Right_Shift()

در این قسمت تابعی تحت عنوان Diagonal_Up_Right_Shift پیاده سازی شده است که تصویر اصلی را به میزان دلخواه شیفت گردشی قطری به بالا و راست می دهد. این تابع علاوه بر تصویر اولیه تعداد پیکسل هایی که قرار است شیفت گردشی به بالا و راست داده شود را نیز دریافت می کند. مثلا اگر بخواهیم به اندازه ۱۰ پیکسل تصویر را به سمت بالا و راست شیفت گردشی دهیم کافی است آرگومان ورودی pixel=10 قرار داده شود. در این دو تابع آرایه با ابعاد تصویر اولیه ایجاد شده و سپس ابتدا دقیقا مشابه آنچه در پیاده سازی Up_Shift بیان شد تصویر را به اندازه pixel به بالا شیفت گردشی داده و خروجی را در متغیر u_shift ذخیره می نماییم سپس دقیقا مشابه آنچه در پیاده سازی Right_Shift بیان شد، تصویر خروجی مرحله قبل یعنی u_shift را به اندازه pixel به راست شیفت گردشی داده و خروجی را در متغیر r_shift ذخیره می نماییم. در انتها ماتریس تصویر نهایی یعنی r_shift به وسیله تابع np.array() به فرمت numpy array تبدیل می شود چرا که تصاویر در این فرمت به عنوان ورودی تابع imshow پذیرفته می شوند.

بلوک [19] فایل ipynb : فراخوانی تابع Diagonal_Up_Right_Shift() و نمایش تصویر خروجی

در این قسمت تابع Diagonal_Up_Right_Shift فراخوانی شده و src_img به عنوان تصویر اولیه و آرگومان pixel با مقدار ۲ به این تابع پاس داده می شوند سپس تصویر حاصل از خروجی تابع و نیز تصویر اولیه به وسیله تابع Show_Images نمایش داده شده است که قابل مقایسه می باشند. دو تصویر نمایش داده شده مشابه شکل (۱۰) می باشد.



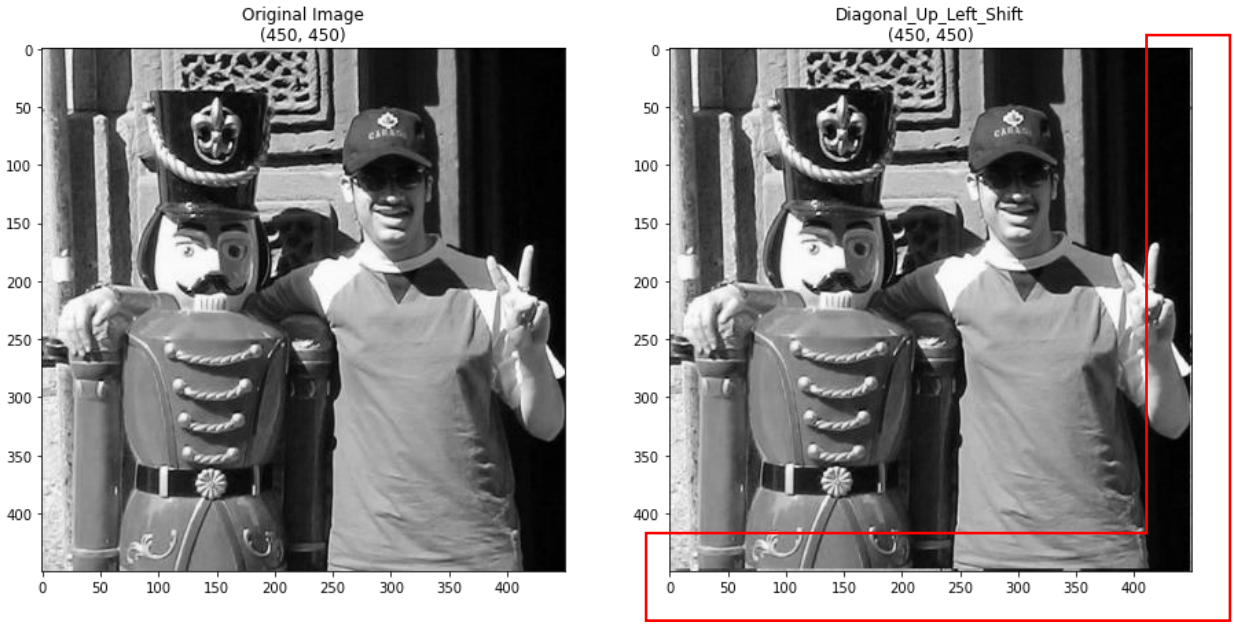
شکل (۱۰)

بلوک [20] فایل ipynb : تابع Diagonal_Up_Left_Shift()

در این قسمت تابعی تحت عنوان Diagonal_Up_Left_Shift پیاده سازی شده است که تصویر اصلی را به میزان دلخواه شیفت گردشی قطری به بالا و چپ می دهد. این تابع علاوه بر تصویر اولیه تعداد پیکسل هایی که قرار است شیفت گردشی به بالا و چپ داده شود را نیز دریافت می کند. مثلا اگر بخواهیم به اندازه ۱۰ پیکسل تصویر را به سمت بالا و چپ شیفت گردشی دهیم کافی است آرگومان ورودی pixel=10 قرار داده شود. در این دو تابع آرایه با ابعاد تصویر اولیه ایجاد شده و سپس ابتدا دقیقا مشابه آنچه در پیاده سازی Up_Shift بیان شد تصویر را به اندازه pixel به بالا شیفت گردشی داده و خروجی را در متغیر u_shift ذخیره می نماییم سپس دقیقا مشابه آنچه در پیاده سازی Left_Shift بیان شد، تصویر خروجی مرحله قبل یعنی u_shift را به اندازه pixel به چپ شیفت گردشی داده و خروجی را در متغیر l_shift ذخیره می نماییم. در انتها ماتریس تصویر نهایی یعنی l_shift به وسیله تابع np.array() به فرمت numpy array تبدیل می شود چرا که تصاویر در این فرمت به عنوان ورودی تابع imshow پذیرفته می شوند.

بلوک [21] فایل ipynb : فراخوانی تابع Diagonal_Up_Left_Shift() و نمایش تصویر خروجی

در این قسمت تابع Diagonal_Up_Left_Shift فراخوانی شده و src_img به عنوان تصویر اولیه و آرگومان pixel با مقدار ۲ به این تابع پاس داده می شوند سپس تصویر حاصل از خروجی تابع و نیز تصویر اولیه به وسیله تابع Show_Images نمایش داده شده است که قابل مقایسه می باشند. دو تصویر نمایش داده شده مشابه شکل (۱۱) می باشد.



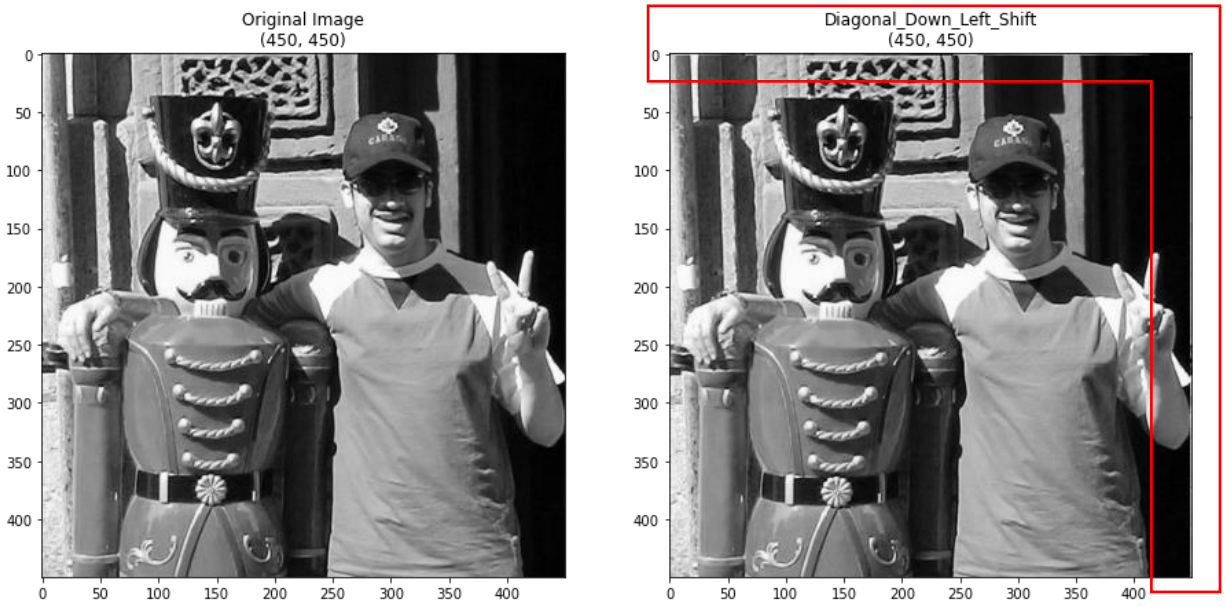
شکل (۱۱)

بلوک [22] فایل ipynb : تابع Diagonal_Down_Left_Shift()

در این قسمت تابعی تحت عنوان Diagonal_Down_Left_Shift پیاده سازی شده است که تصویر اصلی را به میزان دلخواه شیفت گردشی قطری به پایین و چپ می دهد. این تابع علاوه بر تصویر اولیه تعداد پیکسل هایی که قرار است شیفت گردشی به پایین و چپ داده شود را نیز دریافت می کند. مثلا اگر بخواهیم به اندازه ۱۰ پیکسل تصویر را به سمت پایین و چپ شیفت گردشی دهیم کافی است آرگومان ورودی pixel=10 قرار داده شود. در این دو تابع آرایه با ابعاد تصویر اولیه ایجاد شده و سپس ابتدا دقیقا مشابه آنچه در پیاده سازی Down_Shift بیان شد تصویر را به اندازه pixel به پایین شیفت گردشی داده و خروجی را در متغیر d_shift ذخیره می نماییم سپس دقیقا مشابه آنچه در پیاده سازی Left_Shift بیان شد، تصویر خروجی مرحله قبل یعنی d_shift را به اندازه pixel به چپ شیفت گردشی داده و خروجی را در متغیر l_shift ذخیره می نماییم. در انتها ماتریس تصویر نهایی یعنی l_shift به وسیله تابع np.array() به فرمت numpy array تبدیل می شود چرا که تصاویر در این فرمت به عنوان ورودی تابع imshow پذیرفته می شوند.

بلوک [23] فایل ipynb : فراخوانی تابع Diagonal_Down_Left_Shift() و نمایش تصویر خروجی

در این قسمت تابع Diagonal_ Down_Left_Shift فراخوانی شده و src_img به عنوان تصویر اولیه و آرگومان pixel با مقدار ۲ به این تابع پاس داده می شوند سپس تصویر حاصل از خروجی تابع و نیز تصویر اولیه به وسیله تابع Show_Images نمایش داده شده است که قابل مقایسه می باشند. دو تصویر نمایش داده شده مشابه شکل (۱۲) می باشد.



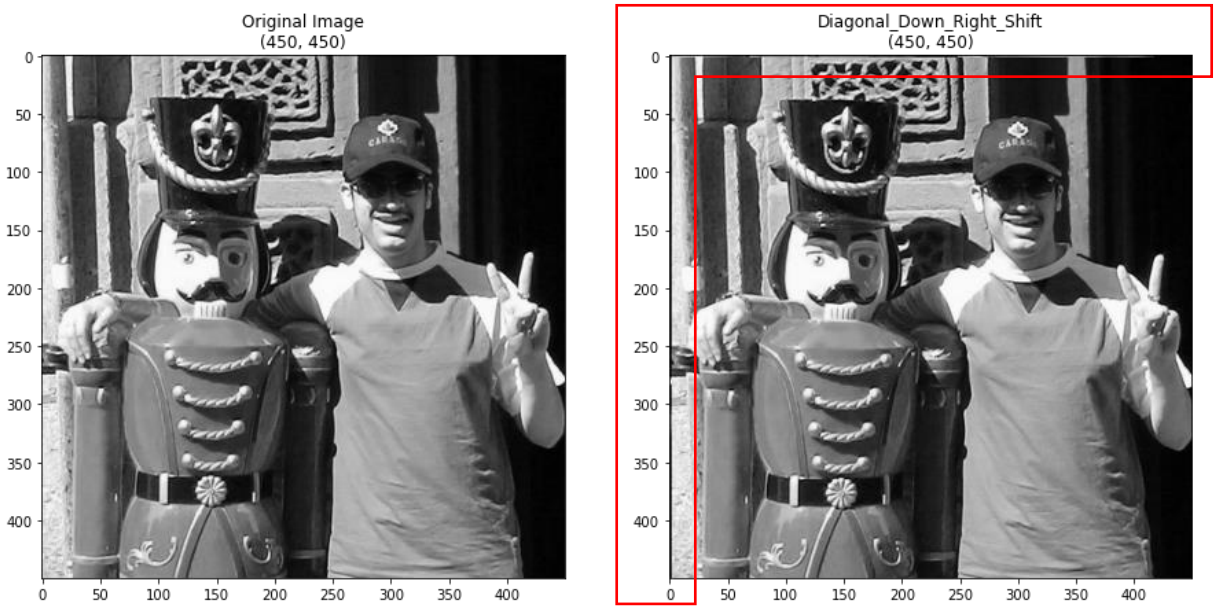
شکل (۱۲)

بلوک [24] فایل ipynb : تابع Diagonal_Down_Right_Shift()

در این قسمت تابعی تحت عنوان Diagonal_Down_Right_Shift پیاده سازی شده است که تصویر اصلی را به میزان دلخواه شیفت گردشی قطری به پایین و راست می دهد. این تابع علاوه بر تصویر اولیه تعداد پیکسل هایی که قرار است شیفت گردشی به پایین و راست داده شود را نیز دریافت می کند. مثلا اگر بخواهیم به اندازه ۱۰ پیکسل تصویر را به سمت پایین و راست شیفت گردشی دهیم کافی است آرگومان ورودی pixel=10 قرار داده شود. در این دو تابع آرایه با ابعاد تصویر اولیه ایجاد شده و سپس ابتدا دقیقا مشابه آنچه در پیاده سازی Down_Shift بیان شد تصویر را به اندازه pixel به پایین شیفت گردشی داده و خروجی را در متغیر d_shift ذخیره می نماییم سپس دقیقا مشابه آنچه در پیاده سازی Right_Shift بیان شد، تصویر خروجی مرحله قبل یعنی d_shift را به اندازه pixel به راست شیفت گردشی داده و خروجی را در متغیر r_shift ذخیره می نماییم. در انتها ماتریس تصویر نهایی یعنی r_shift به وسیله تابع np.array() به فرمت numpy array تبدیل می شود چرا که تصاویر در این فرمت به عنوان ورودی تابع imshow پذیرفته می شوند.

بلوک [25] فایل ipynb : فراخوانی تابع Diagonal_Down_Right_Shift() و نمایش تصویر خروجی

در این قسمت تابع Diagonal_Down_Right_Shift فراخوانی شده و src_img به عنوان تصویر اولیه و آرگومان pixel با مقدار ۲ به این تابع پاس داده می شوند سپس تصویر حاصل از خروجی تابع و نیز تصویر اولیه به وسیله تابع Show_Images نمایش داده شده است که قابل مقایسه می باشند. دو تصویر نمایش داده شده مشابه شکل (۱۳) می باشد.



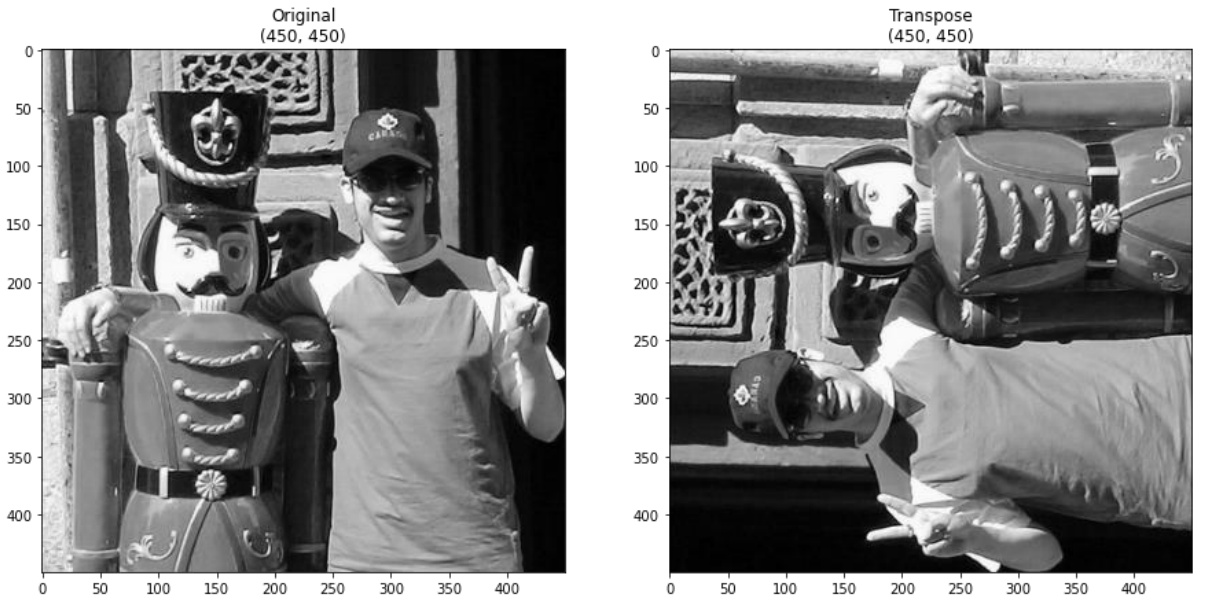
شکل (۱۳)

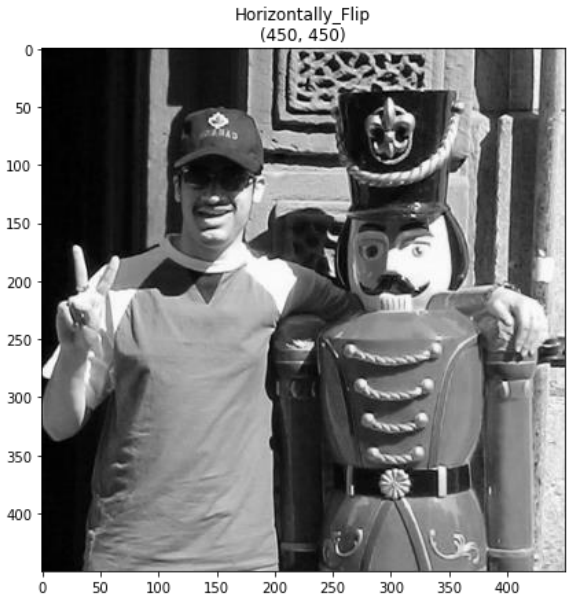
بلوک [26] فایل ipynb : تابع Q1_Function()

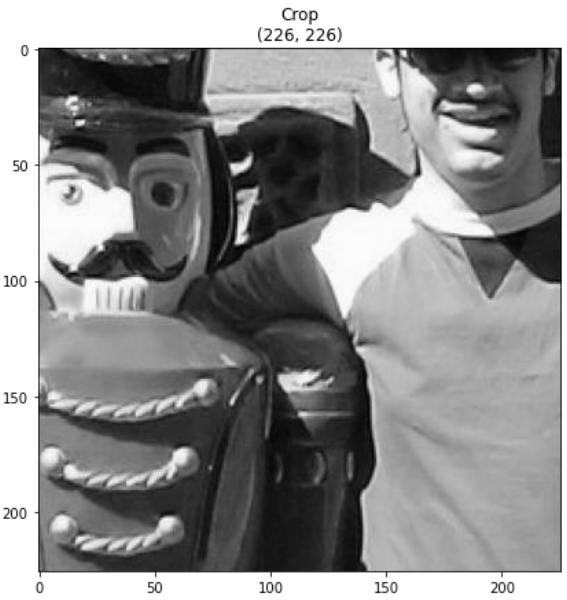
در این قسمت تابعی تحت عنوان Q1_Function به منظور انجام عملیات های ذکر شده در صورت سوال ۱ نوشته شده است که تمام توابع موجود در قسمت های قبلی را به نحوی که خواسته شده است فراخوانی کرده و تصاویر خروجی حاصل از توابع را با کمک تابع Show_Images نمایش می دهد.

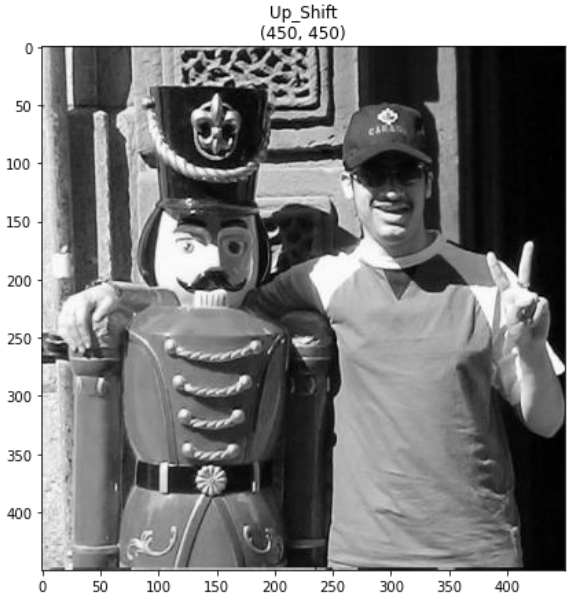
بلوک [27] فایل ipynb : فراخوانی تابع Q1_Function()

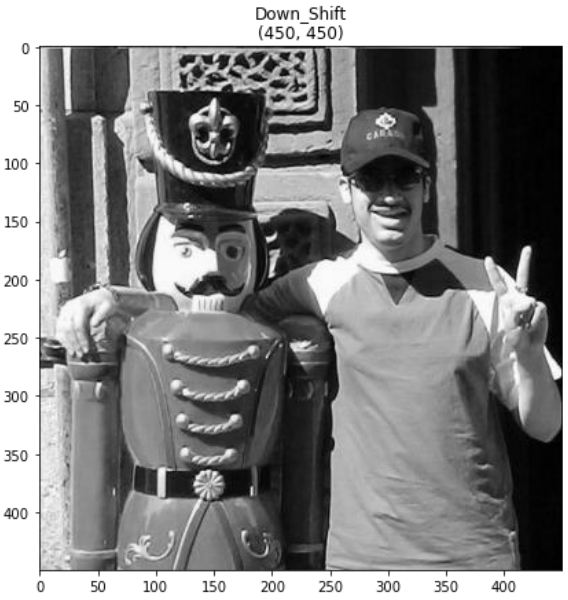
در این قسمت تابع Q1_Function فراخوانی شده و تصاویر مورد نظر به صورت یکجا نمایش داده می شود. خروجی این تابع مطابق با شکل زیر می باشد.

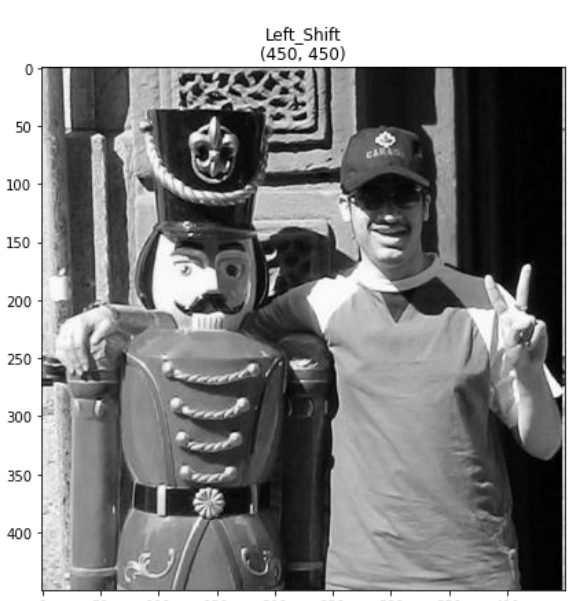


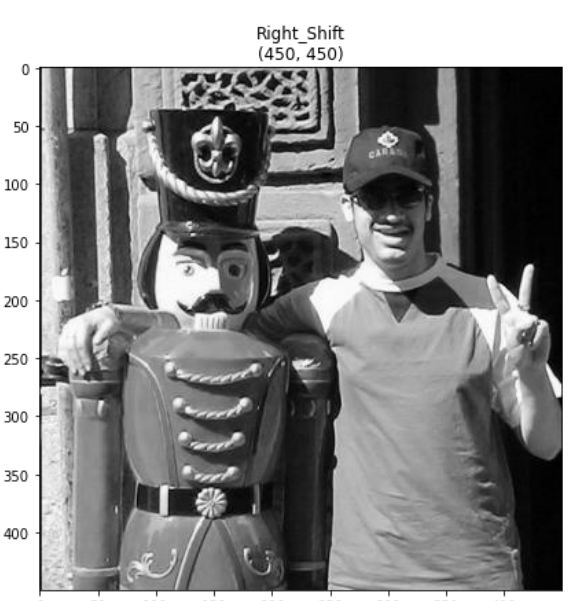

Horizontally_Flip
(450, 450)

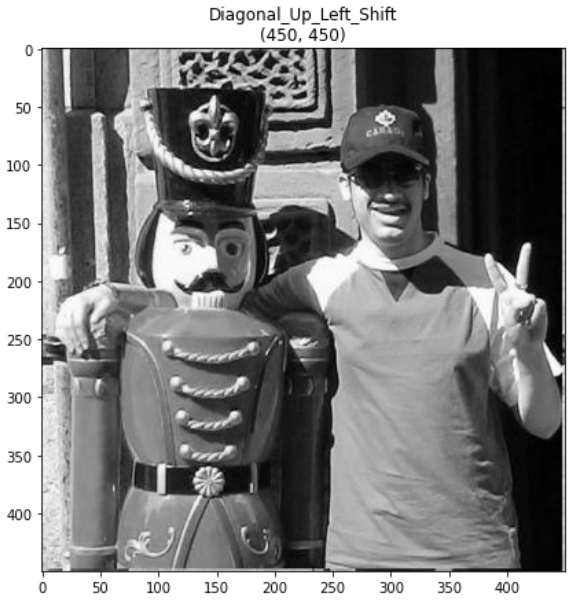

Crop
(226, 226)

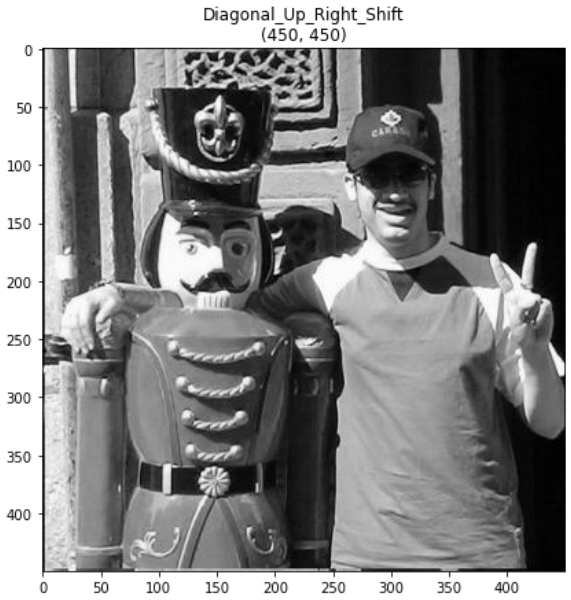

Up_Shift
(450, 450)

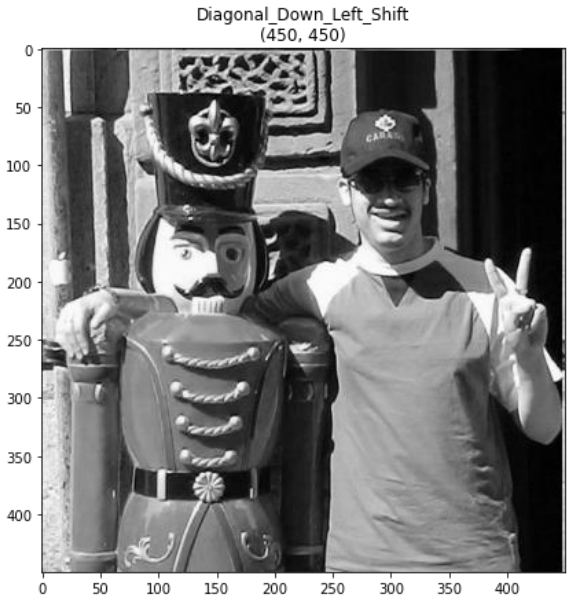

Down_Shift
(450, 450)

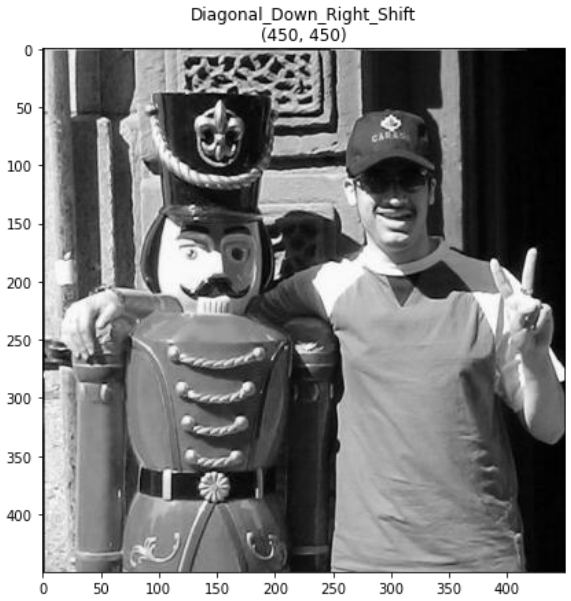

Left_Shift
(450, 450)


Right_Shift
(450, 450)









بلوک [28] فایل ipynb : تابع HW1_MSE()

در این قسمت تابعی تحت عنوان HW1_MSE برای بدست آوردن MSE دو تصویر پیاده سازی شده است. برای پیاده سازی این تابع کافی است تک تک پیکسل های دو تصویر از یکدیگر کم شده و حاصل به توان ۲ رسانده شود سپس مجموع تمام مربعات بدست آمده بر تعداد کل پیکسل های یک تصویر تقسیم گردد. لازم به ذکر است که ابعاد دو تصویر می بایست یکسان باشد.

```
mse = (np.square(imageA.astype(int) - imageB.astype(int))).mean()
```

نکته قابل ملاحظه این است که پیکسل های تصاویر از نوع داده unit8 هستند و لذا برای جلوگیری از سرریز ضمن عملیات تفریق ابتدا نوع داده آن ها را به int تبدیل کرده و سپس تفاضل را صورت می دهیم.

بلوک [29] فایل ipynb : تابع Change_Brightness()

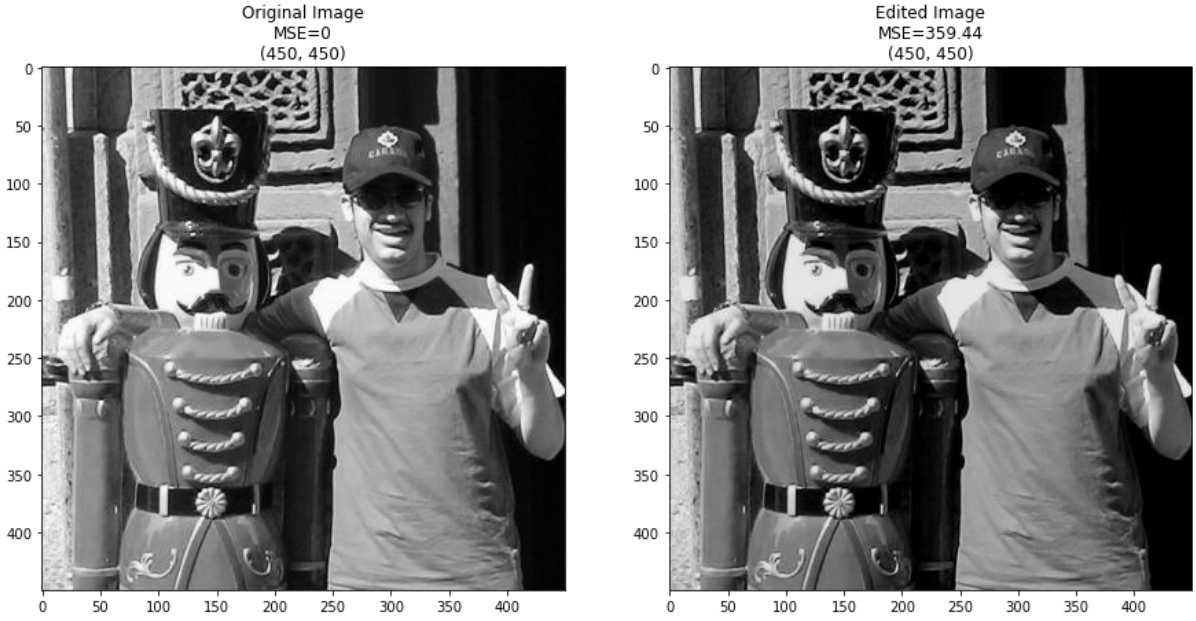
در این قسمت تابعی تحت عنوان Change_Brightness به منظور کاهش روشنایی تصویر به مقدار دلخواه پیاده سازی شده است. این تابع علاوه بر src_img که تصویر ورودی است یک آرگومان level را نیز دریافت می نماید که بیانگر میزان کاهش روشنایی مطلوب است. مثلا اگر بخواهیم روشنایی تصویر ورودی را به اندازه ۱۰ سطح کم کنیم در اینصورت level=10 قرار داده می شود. به علاوه پیش از کاهش مقدار هر پیکسل چک می شود که مقدار اولیه پیکسل مورد نظر از level بزرگتر باشد در غیر اینصورت ضمن تفریق عدد منفی بدست می آید که مطلوب نیست و می بایست به جای آن صفر لحاظ گردد. در ابتدا یک کپی از تصویر اولیه تحت عنوان new_img ذخیره شده و تغییرات در گام های بعدی بر روی این تصویر صورت می پذیرد.

بلوک [30] فایل ipynb : فراخوانی تابع Change_Brightness()

در این قسمت تابع Change_Brightness فراخوانی شده و src_img به عنوان تصویر اولیه و آرگومان level با مقدار ۲۰ به این تابع پاس داده می شوند سپس تصویر حاصل از خروجی تابع و نیز تصویر اولیه به وسیله تابع Show_Images نمایش داده شده است که قابل مقایسه می باشند. دو تصویر نمایش داده شده مشابه شکل (۱۵) می باشد و مقدار MSE تصویر خروجی ۳۵۹/۴۴ بدست آمده است. با توجه به تعریف MSE در نگاه اول به نظر میرسد تعداد هر پیکسل به اندازه ۲۰ واحد کسر شده است پس اختلاف میان هر دو پیکسل تصویر اولیه و تصویر نهایی ۲۰ خواهد بود این مقدار مربع شده و عدد ۴۰۰ بدست می آید به علاوه تعداد کل پیکسل ها ۴۵۰ * ۴۵۰ است و لذا MSE نهایی به صورت زیر مورد انتظار خواهد بود:

$$\frac{20^2 * 450 * 450}{450 * 450} = 400$$

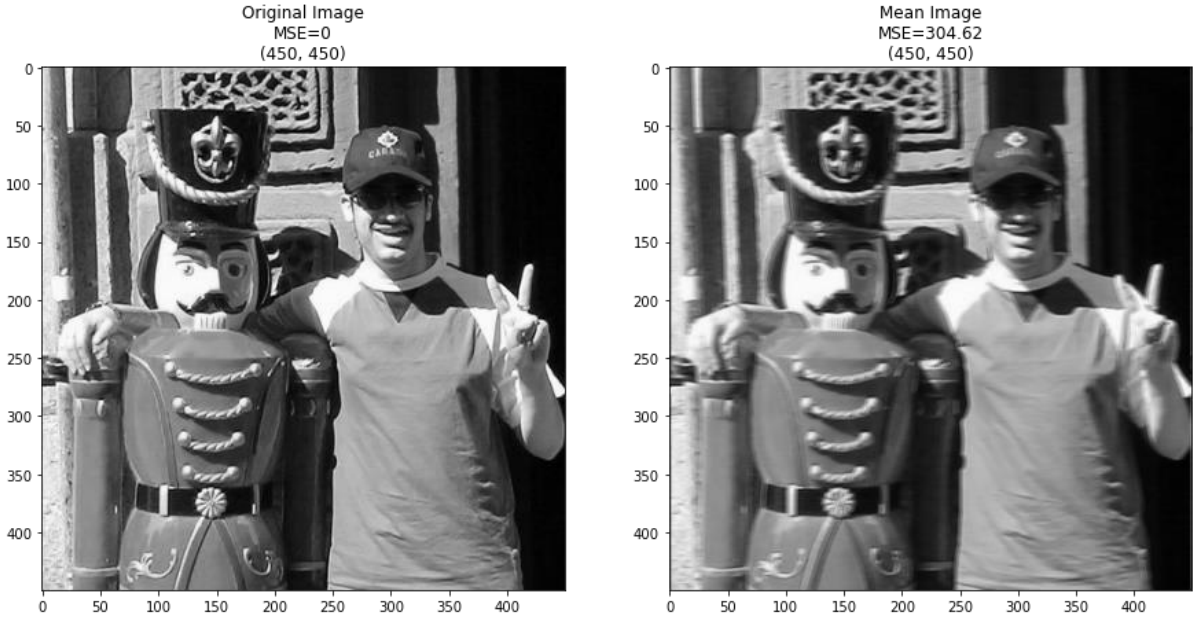
واضح است که مقدار MSE بدست آمده کمتر از میزان مورد انتظار بوده است چراکه برخی از پیکسل ها مقدار اولیه کمتر از ۲۰ داشته اند و لذا ضمن تفاضل به جای مقدار منفی، مقدار ثانویه آن ها صفر لحاظ شده است به همین دلیل در محاسبات فوق همه ۴۵۰ * ۴۵۰ پیکسل مدنظر، اختلاف ۲۰ نخواهند داشت و تعدادی اختلاف کمتر از ۲۰ دارند لذا MSE واقعی از ۴۰۰ کمتر خواهد بود.



شکل (۱۵)

بلوک [31] فایل ipynb : تصویر حاصل از میانگین تصویر اولیه، خروجی Right_Shift و خروجی Left_Shift

در این قسمت ابتدا خروجی توابع Left_Shift و Right_Shift بدست آمده و در متغیر های l_shift و r_shift ذخیره می گردد. سپس پیکسل های هریک از تصاویر حاصله با تغییر نوع داده از uint8 به int تبدیل شده و جمع زده می شوند. علت این امر آن است که ضمن جمع زدن مقدار پیکسل ها، عمل سرریز رخ ندهد و پیکسل های تصویر میانگین به درستی محاسبه گردد. سپس جمع پیکسل های نظیر به نظیر بر ۳ تقسیم می گردد. به علاوه پیکسل های حاصل از میانگین ۳ تصویر به وسیله تابع np.round رند می‌شوند به این دلیل که پیکسل ها با مقادیر اعشاری مجاز نیستند. در انتها نیز مجددا پیکسل های تصویر میانگین به نوع داده uint8 تبدیل می شود تا این تصویر توسط تابع imshow قابل نمایش باشد. MSE بدست آمده برای تصویر نهایی برابر با ۳۰۴/۶۲ است و تصویر حاصل و نیز تصویر اولیه به وسیله تابع Show_Images نمایش داده شده است که قابل مقایسه می باشند. دو تصویر نمایش داده شده مشابه شکل (۱۶) می باشد.



شکل (۱۶)

بلوک [32] فایل ipynb : تابع ()Resize

در این قسمت تابعی تحت عنوان Resize به منظور تغییر ابعاد تصویر نوشته شده است. این تابع علاوه بر src_img که تصویر ورودی است دو آرگومان interpolation و scale_percent را نیز دریافت می‌نماید که به ترتیب بیانگر روش تغییر ابعاد و درصد تغییرات ابعاد است. مثلا اگر بخواهیم ابعاد تصویر ورودی را به اندازه ۱۰ درصد کاهش دهیم یا به عبارت دیگر تصویری با ابعاد ۹۰ درصد تصویر اولیه ایجاد نماییم می بایست scale_percent=0.9 قرارداده شود و همچنین اگر بخواهیم از روش Linear برای تغییر ابعاد استفاده نماییم در اینصورت interpolation=cv2.INTER_LINEAR قرار داده می شود. در بدنه تابع ابتدا طول و عرض تصویر جدید با ضرب کردن درصد تغییرات در طول و عرض تصویر اولیه، بدست آمده و سپس تصویر مورد نظر با روش داده شده و ابعاد جدید ایجاد می گردد و به عنوان خروجی تابع بازگردانده می شود.

بلوک [33] فایل ipynb : فراخوانی تابع ()Q4_Function

در این قسمت تابعی تحت عنوان Q4_Function به منظور انجام عملیات های ذکر شده در صورت سوال ۴ نوشته شده است که از تابع Resize نوشته شده در قسمت پیشین استفاده می نماید و ۳ تصویر هر یک به اندازه ۰/۸ تصویر اولیه با استفاده از سه روش Linear, Cubic, Nearest ایجاد می نماید سپس هر خروجی را با استفاده از همان روش قبلی این بار به اندازه ۱/۲۵ برابر می کند تا به ابعاد تصویر اولیه دست یابیم و سپس MSE نظیر هر خروجی را نسبت به تصویر اولیه محاسبه کرده و تصاویر حاصل از دوبار resize شدن را نمایش می دهد.

بلوک [34] فایل ipynb : فراخوانی تابع Q4_Function()

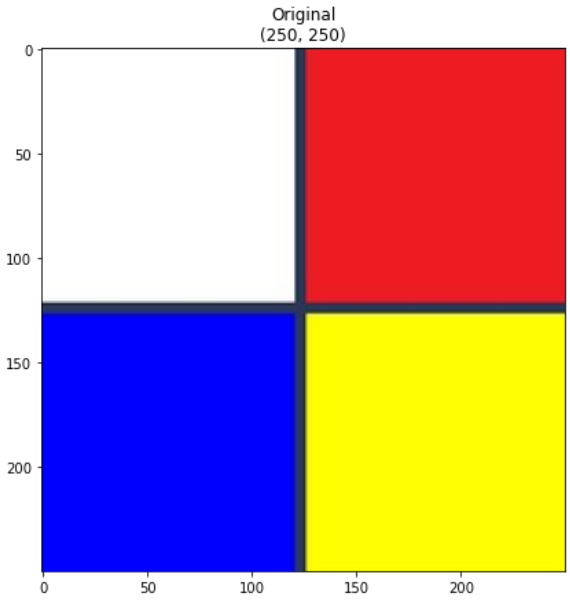
در این قسمت تابع Q4_Function فراخوانی شده و تصاویر خروجی مورد نظر به صورت یکجا نمایش داده می شود. خروجی این تابع مطابق با شکل زیر می باشد.

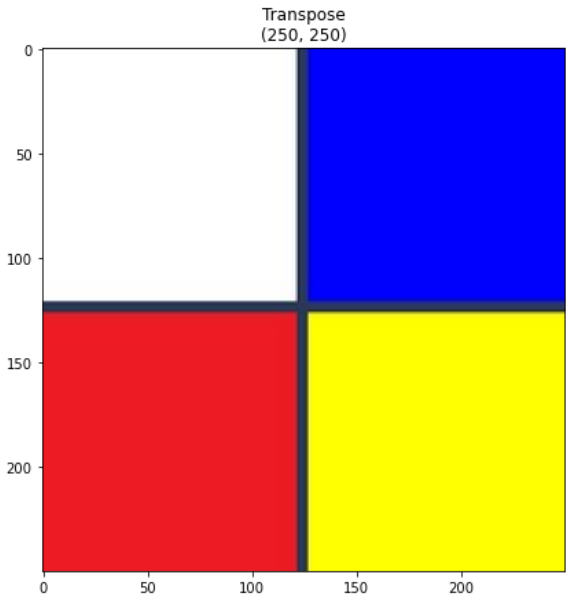
نتایج بدست آمده نشان می دهد اولاً زمانی که یک تصویر با کاهش ابعاد مواجه می گردد بخشی از اطلاعات مربوط به پیکسل ها دور ریخته می شود و حتی در صورتی که مجددا با تغییر بعد به ابعاد اولیه بازگردد دقیقا تصویر اولیه بدست نمی آید چراکه اطلاعات دور ریخته شده ضمن کاهش بعد دیگر قابل بازیابی نیستند. از میان سه روش ذکر شده برای کاهش ابعاد روش Cubic به طور قابل ملاحظه ای بهتر از دو روش دیگر عمل می کند و تصویر بدست آمده از این روش به تصویر اولیه بسیار نزدیک تر است به گونه ای که MSE تصویر

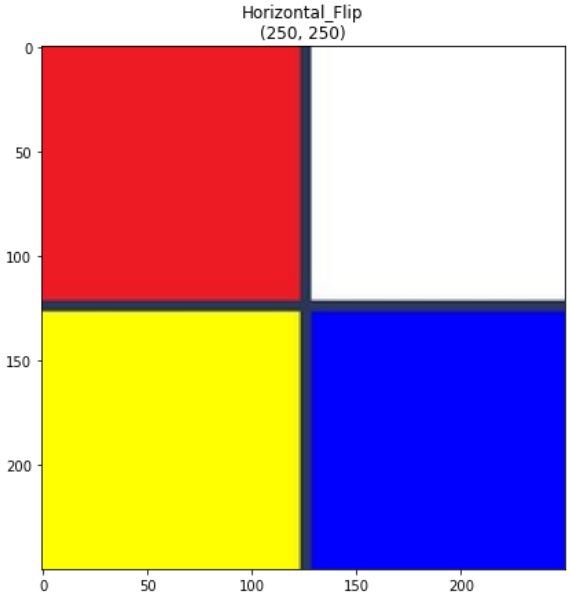
برابر 18.4 شده است. در حالی که روش Nearest بدتر از دو روش دیگر عمل می کند و تصویر بدست آمده از این روش دارای MSE=592.27 است که مقدار بزرگی بوده و حدودا ۳۲ برابر MSE بدست آمده از روش Cubic و ۹ برابر MSE بدست آمده از روش Linear است. روش Linear نیز از نظر عملکرد میان دو روش Cubic و Nearest قرار می گیرد و MSE=63.08 را به ما می دهد.

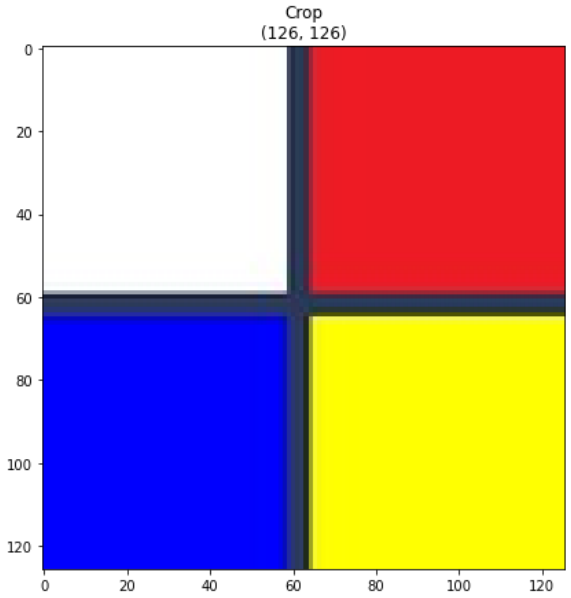
نکته قابل ملاحظه این است که روش های ذکر شده با استفاده از زبان متلب نیز تست گردید و MSE های بدست آمده از خروجی توابع پیاده سازی شده در متلب بسیار کمتر می باشد. به نظر می رسد این توابع در پایتون و متلب به گونه متفاوتی پیاده سازی شده اند و دقت تغییر تصاویر با توابع متلب دقیق تر از پایتون است.

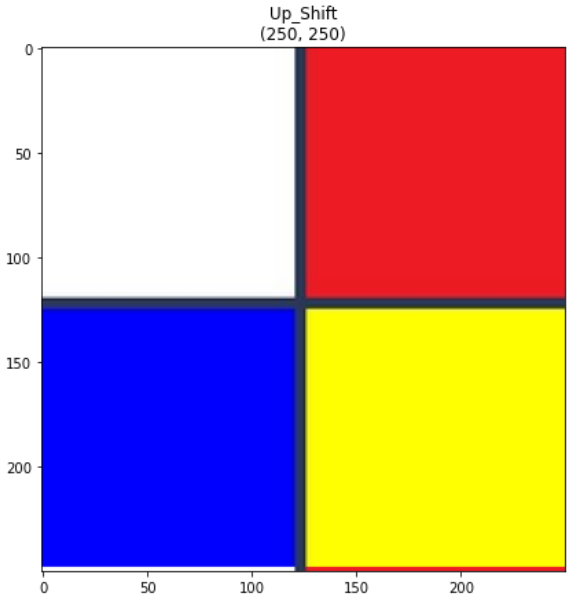
قابل ذکر است که تمام توابع به گونه ای پیاده سازی شده اند که علاوه بر تصاویر سیاه و سفید و مربعی شکل، تغییرات خواسته شده را برروی تصاویر رنگی با هر ابعاد دلخواه نیز پیاده سازی می کنند. کافی است مسیر هر تصویر دلخواه را در img_path قرار دهید و برنامه را از ابتدا اجرا کنید تا تغییرات مطلوب بر روی تصویر اعمال شده و خروجی ها نمایش داده شود. نتایج حاصل از اعمال توابع بر روی یک مورد تصویر رنگی تست شده نیز در ادامه آورده شده است.

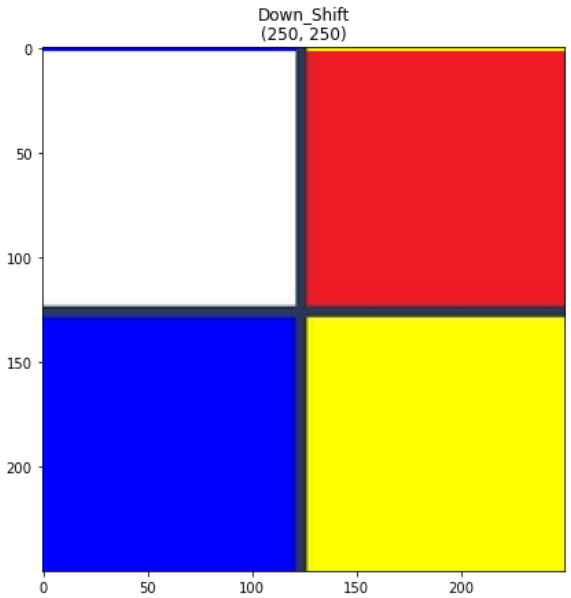

Original
(250, 250)

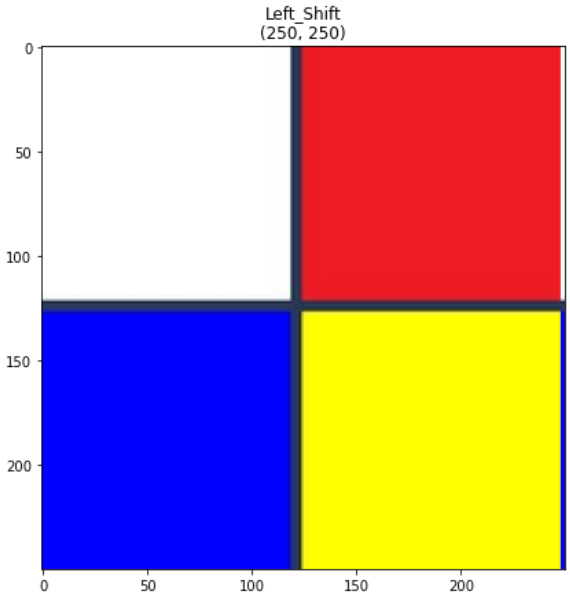

Transpose
(250, 250)

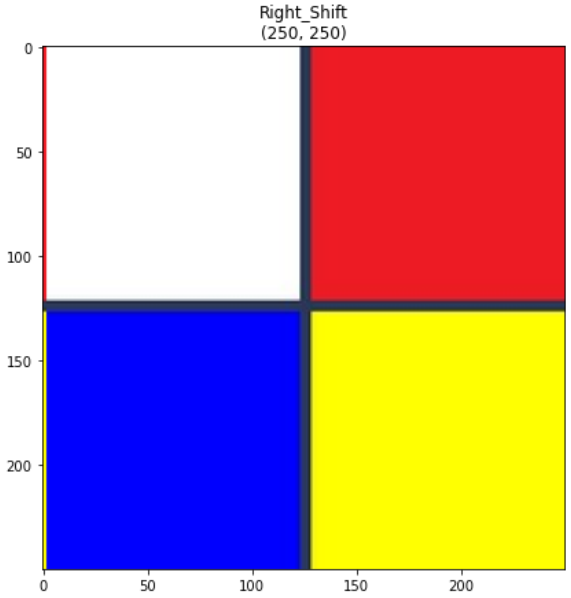

Horizontal_Flip
(250, 250)

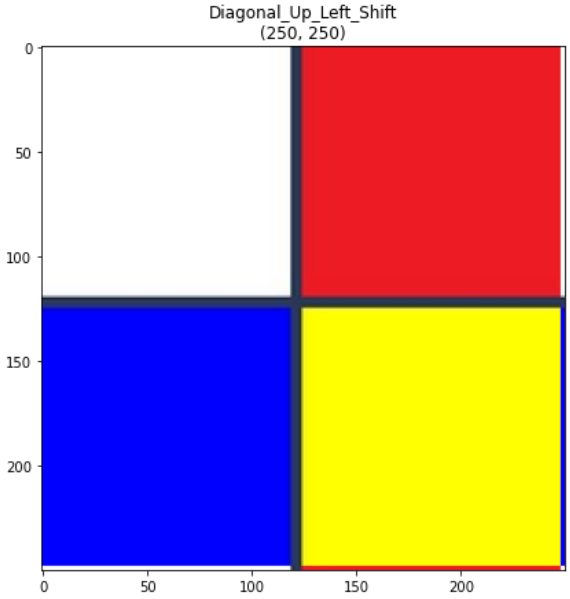

Crop
(126, 126)

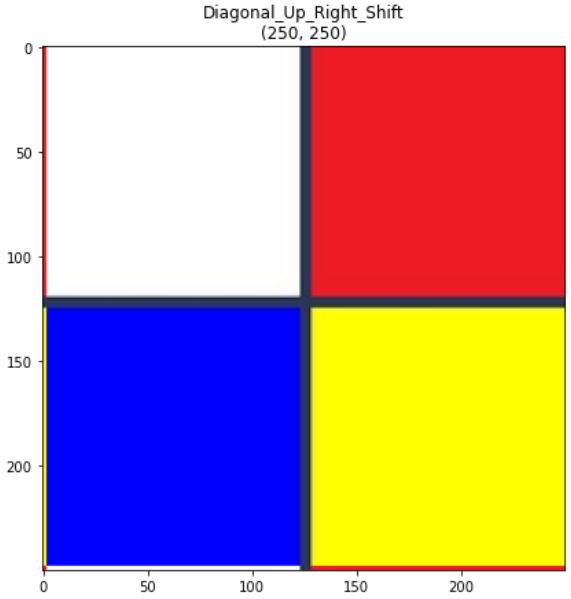

Up_Shift
(250, 250)

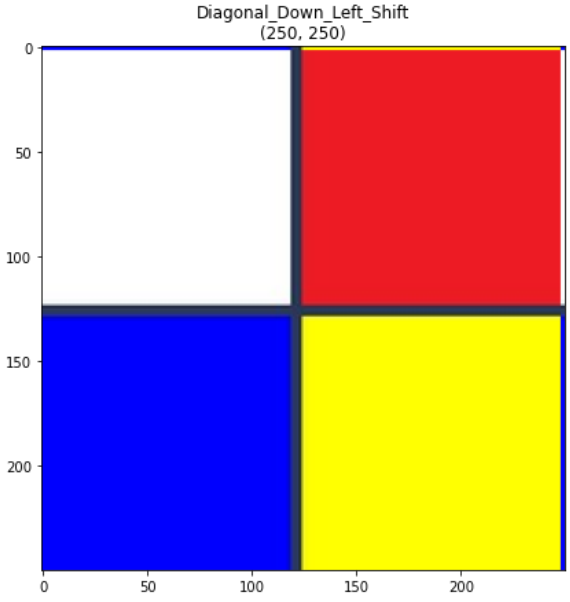

Down_Shift
(250, 250)

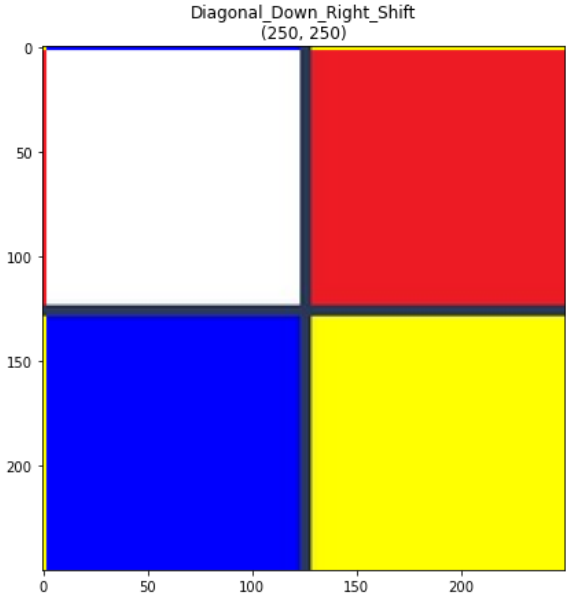

Left_Shift
(250, 250)
0
50
100
150
200
0
50
100
150
200

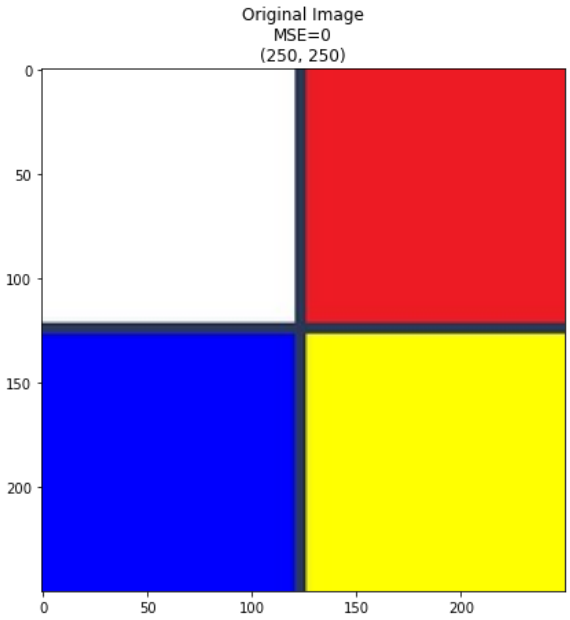

Right_Shift
(250, 250)
0
50
100
150
200
0
50
100
150
200

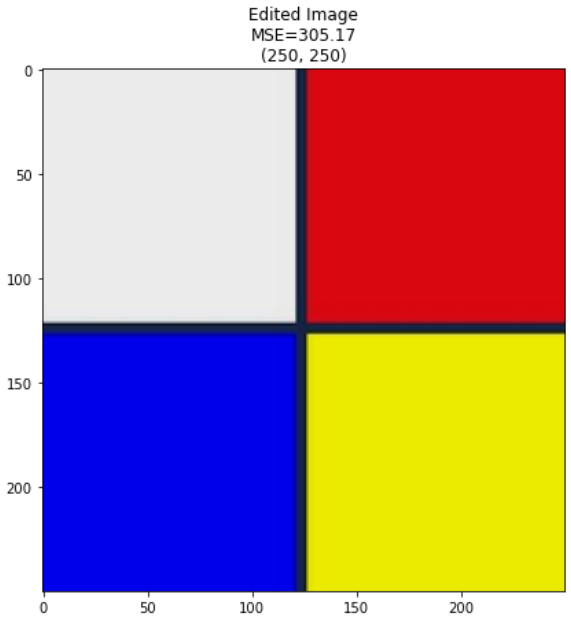

Diagonal_Up_Left_Shift
(250, 250)
0
50
100
150
200
0
50
100
150
200

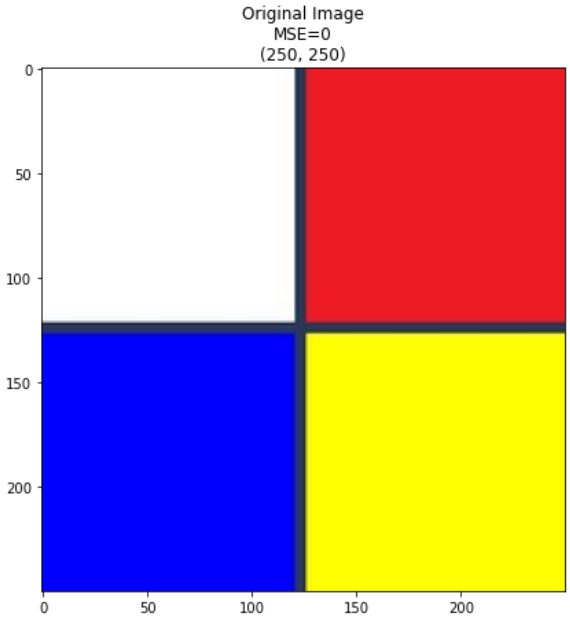

Diagonal_Up_Right_Shift
(250, 250)
0
50
100
150
200
0
50
100
150
200

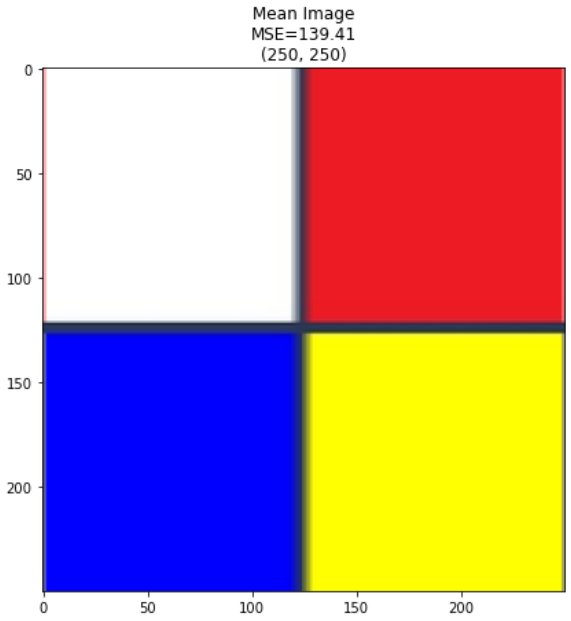

Diagonal_Down_Left_Shift
(250, 250)
0
50
100
150
200
0
50
100
150
200

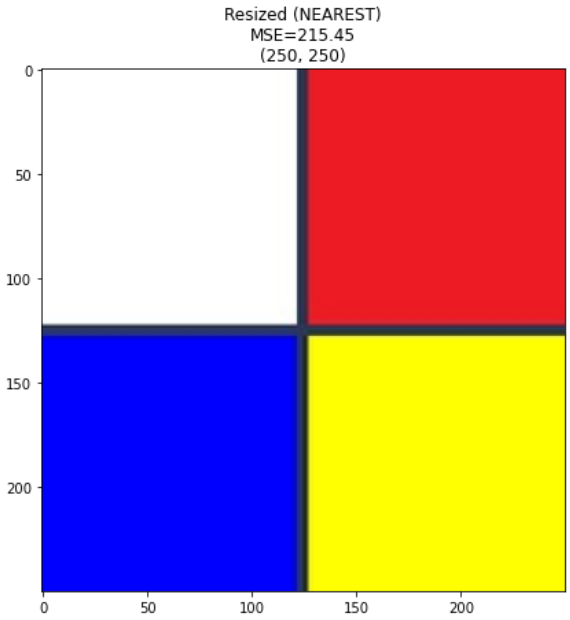

Diagonal_Down_Right_Shift
(250, 250)
0
50
100
150
200
0
50
100
150
200

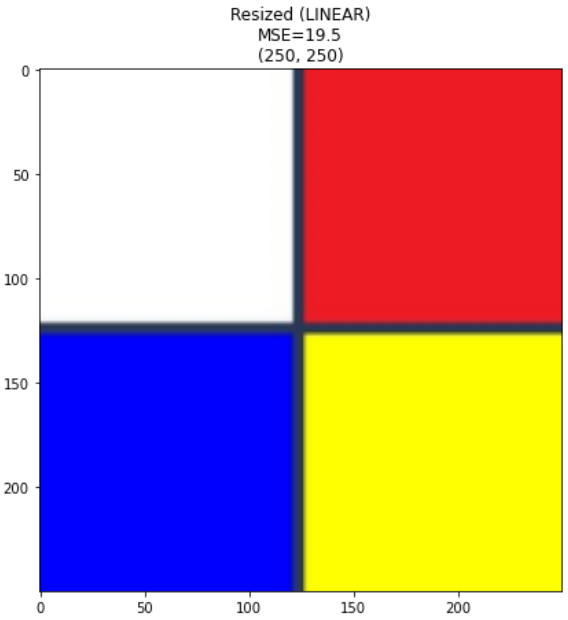

Original Image
MSE=0
(250, 250)
0
50
100
150
200
0
50
100
150
200

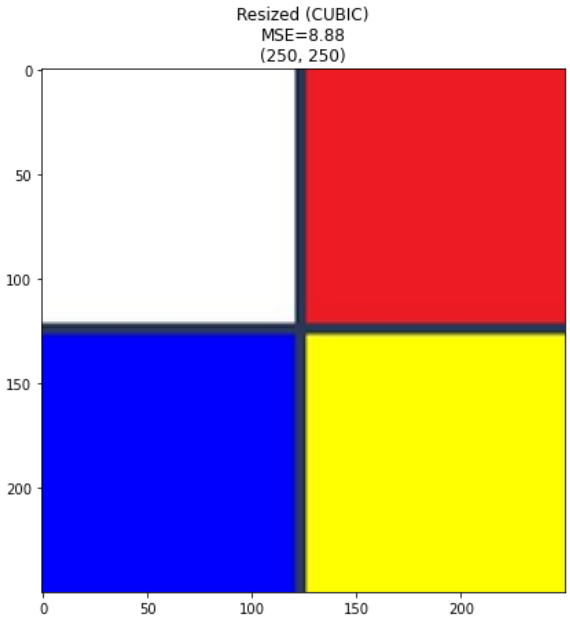

Edited Image
MSE=305.17
(250, 250)
0
50
100
150
200
0
50
100
150
200

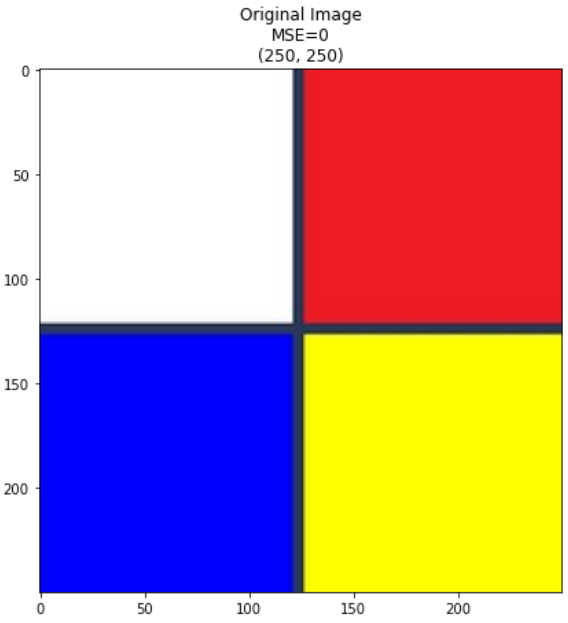

Original Image
MSE=0
(250, 250)
0
50
100
150
200
0
50
100
150
200


Mean Image
MSE=139.41
(250, 250)
0
50
100
150
200
0
50
100
150
200


Resized (NEAREST)
MSE=215.45
(250, 250)
0
50
100
150
200
0
50
100
150
200


Resized (LINEAR)
MSE=19.5
(250, 250)
0
50
100
150
200
0
50
100
150
200


Resized (CUBIC)
MSE=8.88
(250, 250)
0
50
100
150
200
0
50
100
150
200


Original Image
MSE=0
(250, 250)
0
50
100
150
200
0
50
100
150
200