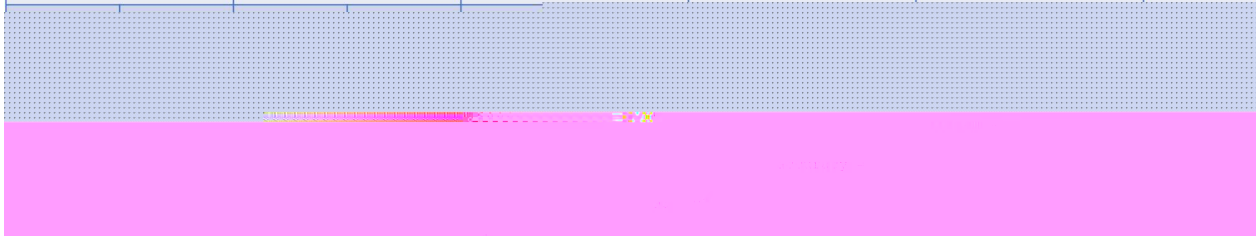


1. اگر کرنلی به اندازه $n*n$ داشته باشیم، می توان گفت که هر پیکسل تصویر n^2 بار visit می شود که به ازاء هر بار visit شدن در یکی از وزن های کرنل ضرب می شود. پس به طور کلی می توان گفت سطح روشنایی هر پیکسل به n^2 قسمت با وزن های $w(1, 1)$ تا $w(n, n)$ در پیکسل های همسایه تقسیم می شود. با جمع کردن سطح روشنایی تمامی پیکسل ها fraction های هر پیکسل مجددا با هم جمع می شوند و با توجه به یک بودن مجموع ضرایب کرنل می توان گفت در انتها از سطح روشنایی هر پیکسل یک عدد موجود است پس مجموعشان نسبت به پیش از اعمال فیلترینگ ثابت می ماند.

2. الگوریتم: اگر سطح روشنایی پیکسل 0 یا 1 نباشد، تغییری ایجاد نمی کنیم. اگر سطح روشنایی 0 یا 1 باشد، میانگین پیکسل های غیر 0 و 1 تحت کرنل محاسبه شده در پیکسل مورد بررسی قرار داده می شود. در صورتی که تمام پیکسل های تحت کرنل نویز تشخیص داده شوند مقدار پیکسل مورد بررسی برابر 0.5 قرار داده می شود. جدول زیر نتایج PSNR با کرنل $3*3$ می باشد. کد مربوطه در فایلی به نام My_denoising.m پیوست شده است.

نام تصویر	نویز 10%	نویز 30%	نویز 50%	نویز 70%	نویز 90%
					

3. کد مربوطه در سه فایل main2.m, My_template_match.m و My_insert_text.m پیوست شده است. تصاویر خروجی و نمونه های استفاده شده برای template matching و insert_text به ترتیب در فولدر هایی به نام های result و sample و insert_text پیوست شده اند. دقت نهایی 95٪ می باشد.



تصاویر دارای خطا: 47 و 68 و 70 و 75 و 76
75: عدم تشخیص 6 کوچک، بقیه تصاویر: عدم تشخیص 4 کوچک

الگوریتم: ابتدا تصاویر با استفاده از MF، denoise شده و به تصاویر $970*970$ تبدیل شده اند و در فولدری دیگر ذخیره شده اند که از اجرای این بخش در هر بار تست کردن اجتناب کنیم. سپس در یک حلقه تابع My_template_match برای هر تصویر اجرا می شود و سپس خروجی تابع روی تصویر اصلی نوشته شده و در صورت صحیح بودن عدد مربوطه یک واحد به accuracy اضافه می شود. در تابع My_template_match به ازاء هر عدد 1 تا 8 تصویر را پیمایش کرده و روی نقاطی که سفید نباشند تصاویر نمونه در سه سایز مختلف قرار داده شده و psnr

محاسبه می شود. در صورتی که حاصل psnr از 11 بیشتر باشد، match رخ داده و عدد مربوطه به result اضافه می شود. همچنین برای جلوگیری از تشخیص مجدد این عدد ناحیه ای که در آن match اتفاق افتاده سفید می شود. در تابع My_insert_text نمونه هایی از اعداد 0 تا 9 بارگذاری شده و تصویر ورودی در سه بخش R و G و B تصویر خروجی کپی می شود. سپس عدد مربوطه با نمونه هایی که در اختیار تابع قرار گرفته ساخته می شود به این صورت که در مکانی که نمونه رنگی غیر از سفید دارد، G و B تصویر خروجی را صفر می کنیم تا رنگ قرمز برای آن نقطه به دست آید. این عملیات یک بار برای دهگان و یک بار برای یکان عدد ورودی انجام می شود.