به نام خدا

**عنوان**

بخش سوم از تکلیف اول درس پردازش تصویر رقمی

**استاد**

دکتر منصوری

**دانشجو**

محمدعلی مجتهدسلیمانی

4033904504

**تاریخ**

5/02/1404

Table of Contents

[**سوال 5** 3](#_Toc196514217)

[**بخش الف** 3](#_Toc196514218)

[**فیلترهای تقویتی** 3](#_Toc196514219)

[**فیلترهای هموار ساز** 4](#_Toc196514220)

# **سوال 5**

این سوال در 3 بخش حل شده است که قسمت هایی که نیاز به پیاده سازی دارند در گزارش کار بخش مربوطه آن توضیحات پیاده سازی آمده است.

## **بخش الف**

در این بخش تفاوت بین 2 دسته از فیلترهای تقویتی و فیلترهای هموارسازی را بررسی میکنیم:

### **فیلترهای تقویتی**

هدف این فیلتر ها بهبود جزئیات و واضح تر کردن لبه ها یا برجسته کردن آنها است. در واقع هدف آنها افزایش کنتراست محلی در امتداد لبه ها است. فیلترهای تقویتی معمولا با تاکید بر تفاوت بین مقادیر پیکسل مجاور کار میکنند. روش های رایج عبارتند از:

**روش unsharp masking**: این روش که یکی از رایج ترین روش ها است که 3 مرحله دارد: اول یک نسخه تار (blur) از نسخه اصلی تولید میکند ( اغلب به کمک gaussian blur) دوم نسخه تار را از نسخه اصلی کم میکنند که باعث میشود جزئیات فرکانس بالا مانند لبه ها را جدا کند. سوم افزودن این جزئیات به تصویر اصلی است که به طور موثر کنتراست را در لبه ها افزایش میدهد.

**روش فیلتر لاپلاسین**: این یکی از روش های تقویتی است که بر اساس مشتق دوم کار میکند که نواحی که در آن تغییر شدت بالا و سریع است مانند لبه ها یا نویز را بر جسته میکند. افزودن خروجی لاپلاس به تصویر اصلی این نواحی را بهبود میبخشد.

در نهایت تاثیر این فیلترها لبه ها را برجسته تر میکند و بافت ها را بهبود میبخشد و میتواند تصویر را تازه تر یا صاف تر نشان دهند. با این حال به تقویت نویز موجود نیز کمک میکند.



تصویر پایین بیانگر خروجی فیلتر گرادیان و فیلتر لاپلاسین است:



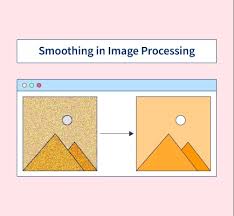
### **فیلترهای هموار ساز**

این فیلتر ها مانند gaussian blur، به دنبال کاهش نویز و جزئیات هستند تا بتوانند ظاهری صاف تر و هموار تر ایجاد کنند. هدف آنها کاهش تفاوت بین پیکسل های مجاور است.

فیلتر های هموار ساز مقدار هر پیکسل را با میانگین وزنی پیکسل های مجاور آن جایگزین میکنند.

**Gaussian blur**: در این روش از یک طرح وزن دهی خاصی بر اساس تابع گوسی استفاده میکند. پیکسل های نزدیک تر به مرکز هسته فیلتر نسبت به پیکسل های دورتر تاثیر بیشتری بر میانگین دارند. این منجر به تاری بسیار هموار و طبیعی در مقایسه با تاری که با کمک میانگین یکنواخت بدست میاد بشود. میزان تاری توسط سیگما تابع گوسی کنترل میشود. سیگما بزرگتر به معنای تاری بیشتر است.

این فیلتر ها باعث میشوند که جزئیات که تیز تر هستند یا تند تر هستند را کاهش دهند و لبه ها را نرم تر کنند. به طور موثر فرکانس های بالا مانند نویز گوسی و نویز نمک و فلفل را کاهش دهند. این باعث میشود تصویر با جزئیات کمتر به نظر برسد.



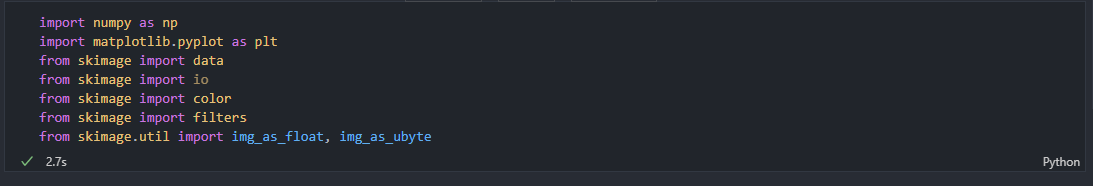
|  |  |  |
| --- | --- | --- |
| فیلترهای هموار ساز | فیلترهای تقویتی | ویژگی |
| کاهش نویز و جزئیات | بهبود لبه ها و جزئیات | هدف |
| میانگین وزن دار همسایگی لحاظ میکند | اهمیت دادن به تغییر مقادیر پیکسل | مکانیزم |
| مانند فیلتر low-pass رفتار میکند | مانند فیلتر high-pass رفتار میکند | فرکانس |
| لبه ها را نرم تر میکنند و کنتراست را کاهش میدهند | باعث میشود لبه ها برجسته تر شوند و کنتراست را زیاد میکند | تاثیر بر لبه |
| جزئیات را کاهش میدهند | باعث افزایش جزئیات میشود | تاثیر بر جزئیات |
| نویز را کاهش میدهند | باعث تقویت نویز میشوند | تاثیر بر نویز |

جدول بالا به طور خلاصه تفاوت این 2 روش را توضیح میدهد.

بخش ب

در این بخش ما از 2 فیلتر high-pass به نام Laplacian و unsharp mask استفاده کردیم که توضیحات هر یک را بالاتر داده ایم، در ادامه به جزئیات پیاده سازی میپردازیم:

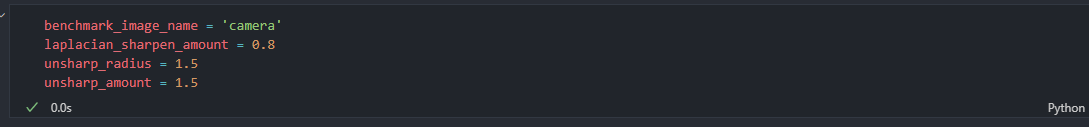
گزارش کار



از matplotlib برای نمایش خروجی استفاده کردیم.

با کمک کتابخانه scikit هم برای تصویر ورودی و هم برای اعمال فیلتر ها و هم برای تبدیل ها استفاده کردیم. میتوانستیم همین کار را با open cv نیز انجام دهیم.

تنظیم پارامتر

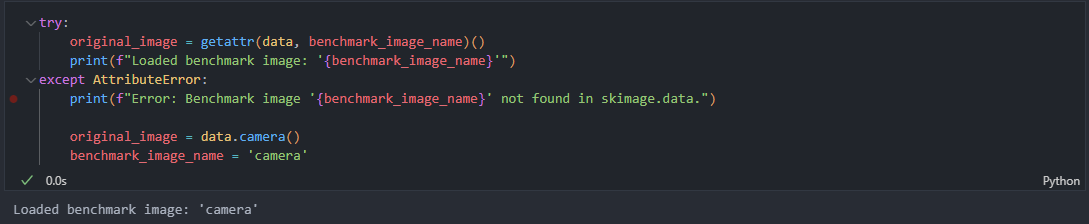


تصویر ورودی را مشخص کردیم به عنوان benchmark.

در ادامه پارامتر لاپلاسین را مشخص کردیم که مشخص میکند چه مقدار از لاپلاسین باید کم شود از تصویر اصلی هر چه این مقدار بزرگتر باشد تصویر تیز تر و برجسته تر خواهد بود.

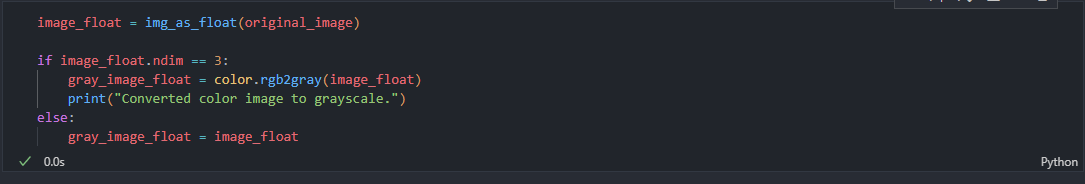
در ادامه پارامتر های فیلتر unsharp را مشخص کردیم. Radius مشخص میکند تخمین سیگما گاوسی را که میزان هموار شدن جزئیات را کنترل میکند. Amount نیز قدرت نواحی تیز و تند را مشخص میکند یعنی به چه میزان باید وزن تخصیص کنیم. هر چه بیشتر باشد تیز تر خواهد بود.

بارگذاری تصویر



در این قسمت صرفا تصویر را بارگذاری کردیم و بررسی کردیم تصویر به درستی کار بکند.

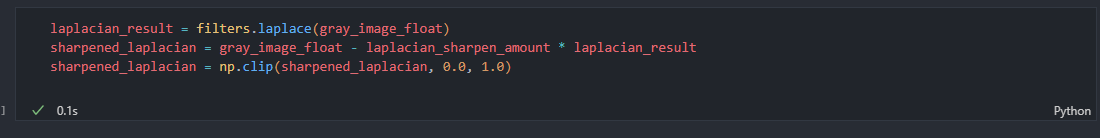
سپس



اول تصویر را به float بازنویسی میکنیم زیرا برای کتابخانه skimage ظاهرا طبق پیاده سازی آن نیاز است.

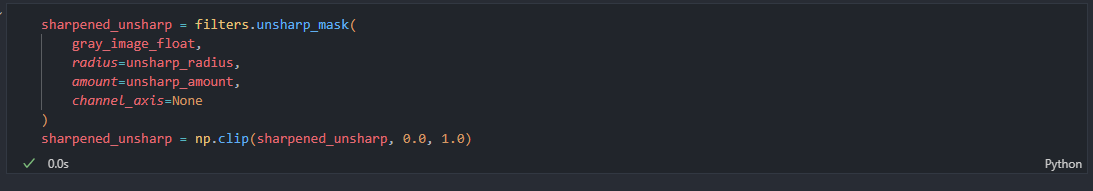
بعد از آن مطمئن میشویم تصویر حتما grayscale است و 3 کاناله نیست.

پیاده سازی اول: لاپلاسین



مقدار لاپلاسین تصویر را حساب میکنیم. لاپلاسین نواحی با تغییرات شدید مانند لبه ها را برجسته میکند. در مرحله بعد با کم کردن لاپلاسین از تصویر اصلی سعی میکنیم تصویر برجسته تر و با جزئیات تر بکنیم. کم کردن مقدار لاپلاسین مثبت باعث میشود وقتی لبه ها تشخیص داده شدند مقدار گرادیان زیاد شود. در نهایت خروجی را با کمک clip محدود میکنیم.

پیاده سازی اول: unsharp masking

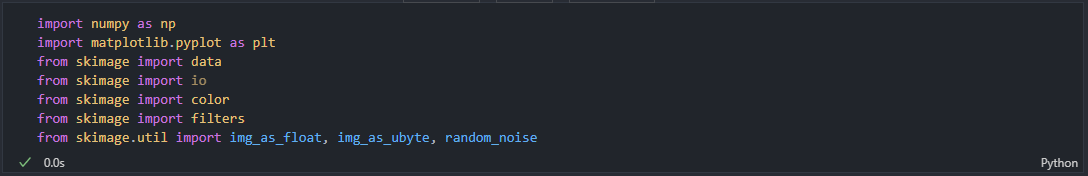


مقایسه

## **بخش ج**

توضیحات مربوط به روش لاپلاسین در بخش الف به طور کامل گفته شده است و اینجا به جزئیات پیاده سازی میپردازیم و نتایج را در نهایت تحلیل میکنیم.

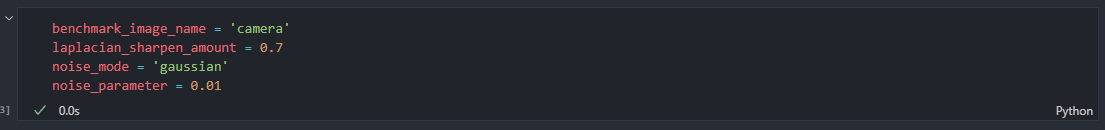
### **گزارش کار**



از matplotlib برای نمایش خروجی استفاده کردیم.

با کمک کتابخانه scikit هم برای تصویر ورودی و هم برای اعمال فیلتر ها و هم برای تبدیل ها استفاده کردیم. میتوانستیم همین کار را با open cv نیز انجام دهیم.

تنظیم پارامتر

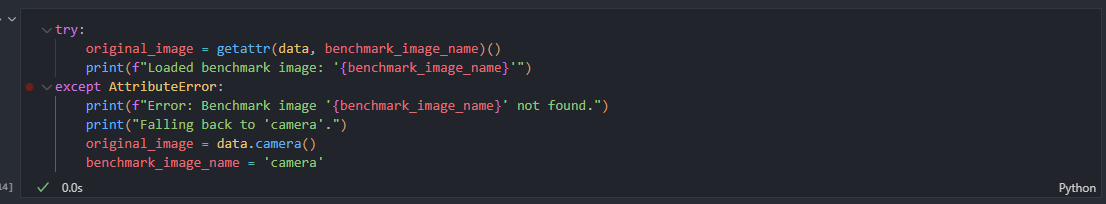


تصویر ورودی را مشخص کردیم به عنوان benchmark.

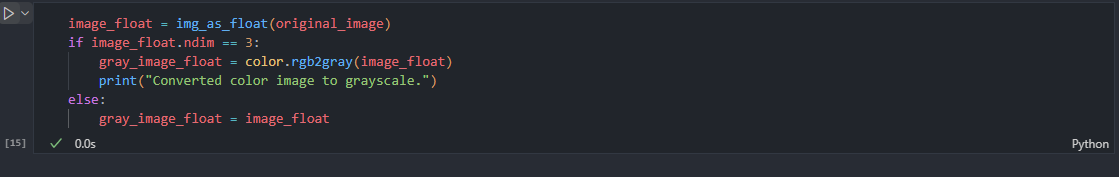
در ادامه پارامتر لاپلاسین را مشخص کردیم که مشخص میکند چه مقدار از لاپلاسین باید کم شود از تصویر اصلی هر چه این مقدار بزرگتر باشد تصویر تیز تر و برجسته تر خواهد بود.

در ادامه نوع نویز را مشخص کردیم میتوانستیم از نویز نمک و فلفل نیز استفاده کنیم.

بعد از با کمک Nosie parameter مقدار واریانس برای نویز گاوسی را مشخص کردیم.

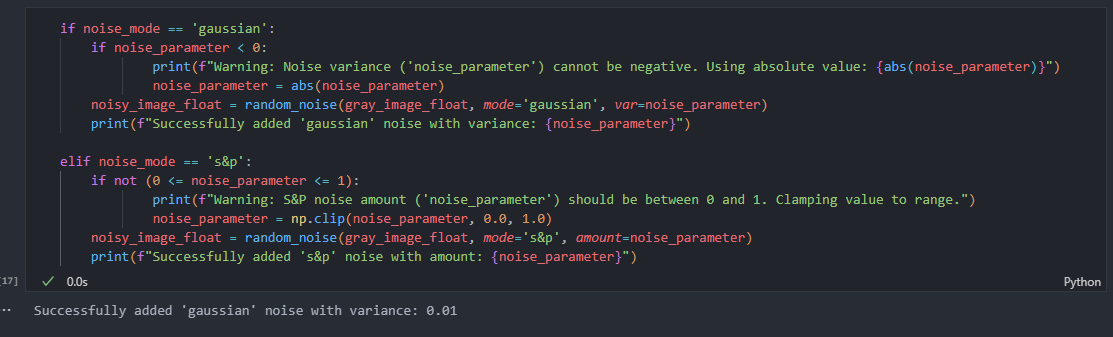


عکس مورد نظر را بارگذاری کردیم و مطمئن شدیم درست کار میکند.

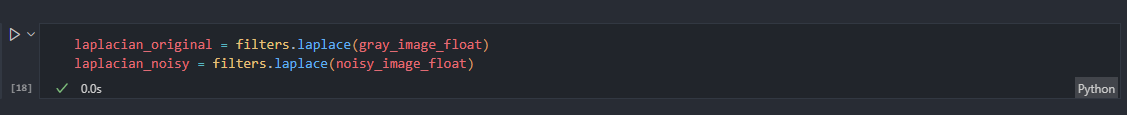


اول تصویر را به float بازنویسی میکنیم زیرا برای کتابخانه skimage ظاهرا طبق پیاده سازی آن نیاز است.

بعد از آن مطمئن میشویم تصویر حتما grayscale است و 3 کاناله نیست.

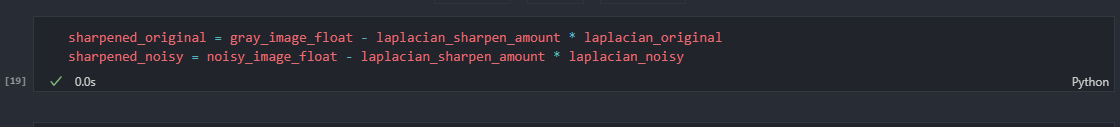


در این مرحله میخواهیم نویز را به تصویر اعمال کنیم. این کار با ساختن یک نسخه نویزی از تصویر با کمک skimage.util.random\_noise انجام میشود. همچنین پیشبینی کرده ایم که اگر کاربر بخواهد از نویز نمک و فلفل استفاده بکند نیز پوشش بدهیم.



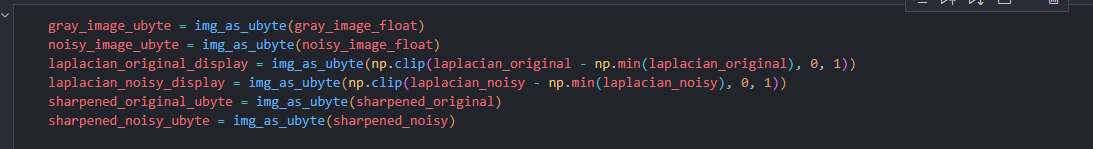
بعد از آن فیلتر لاپلاس را اعمال کردیم.

هم بر روی تصویر اصلی و هم بر روی تصویر نویزی شده، فیلتر را اعمال کرده ایم.

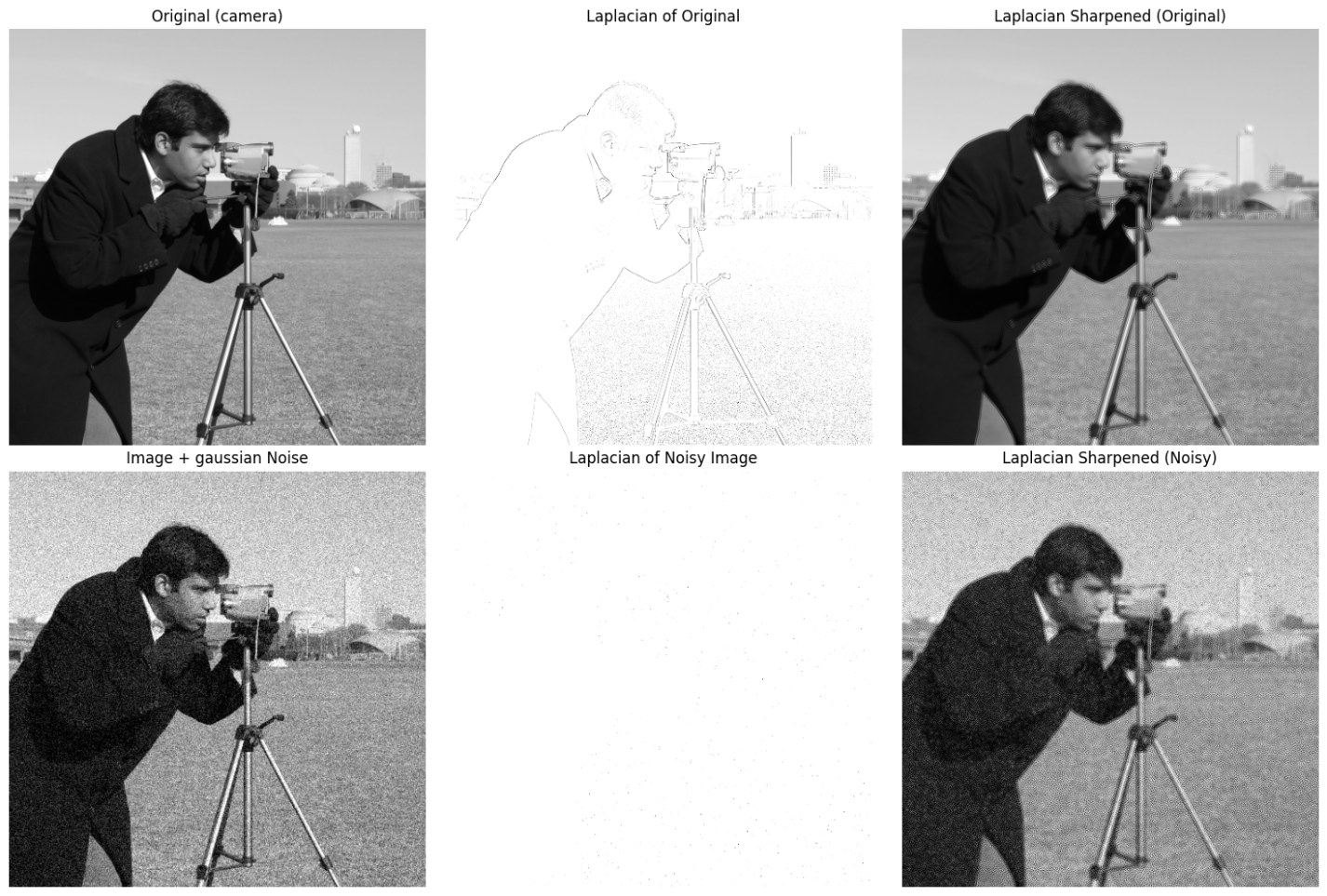


Sharpening را روی تصویر اعمال کردیم.

در نهایت خروجی را به 8 بیت تبدیل کردیم تا قابل نمایش باشد. و بعد با کمک کتابخانه matplotlib آن را نمایش میدهیم.



### **خروجی**



### **تحلیل**

همانطور که در بخش اول گفتیم روش لاپلاسین میتواند به خوبی نقاطی که روشنایی به سرعت تغییر میکنند و نقاط تیز را پیدا کند یعنی همان جزئیات و لبه ها ولی مشکل این است که اینکار همراه با نویز معمولا از نوع speckle همراه است. برای اینکه عکس را تیز تر کنیم یعنی لبه ها را برجسته تر کنیم نیاز داریم یک map از نقاط تیز داشته باشیم و آن را از تصویر اصلی کم کنیم همانطوری که قبلا در بالا توضیح دادیم.

اگر تصویر ورودی بدون نویز باشد همانطور که قابل مشاهده است این روش خوب کار میکند، لبه ها را پیدا میکند و تصویر صاف تر و تیز تر تولید میکند.

اگر تصویر نویزی باشد، چون که ما یک سری نویز اضافه کردیم، این نویزها تیز هستند و در روشنایی تاثیر گذار هستند همانطور که قابل مشاهده است، روش لاپلاسین حساس خواهد شد و نویز ها را هم همانند لبه ها برجسته میکند، و وقتی از نگاشت نقاط تیز استفاده میکنیم، دانه های نویز را همانند لبه های واقعی تقویت میکنیم و وضعیت نویز بدتر میشود و خروجی به طور کلی بدتر میشود.