تمرین سری ۲ تمرین شماره ۱

محمد جواد شریعتی ۹۶۱۰۰۴۱۴

توضیح فایل helper.py

در تمامی تمرین ها،کنار فایل اصلی کد، یک فایل helper.py هم قرار داده شده است. این فایل به منظور تمیزتر شدن کد نوشته شده است. تابع هایی که استفاده میکنم را در این فایل پیاده سازی می کنم تا کد اصلی تمیز تر بشود.

برای بدست آوردن فیلتر مشتق گاوس تابع گاوس دو بعدی را مینویسیم. سپس از آن مشتق میگیریم. برای مثال مشتق گاوس دوبعدی نسبت به x چنین خواهد بود:

$$-\frac{(x-\mu)}{2\pi\sigma^4}\, e^{-\frac{(x-\mu)^2+(y-\mu)^2}{2\sigma^2}}$$

به وضوح مشتق نسبت به y هم به همین صورت بدست می آید. این را در تابع Gaussian_derivative پیاده سازی کرده ام.
سپس از روی این تابع، باید یک فیلتر بسازیم، باامتحان سایز های مختلف، سایز ۷*۷ بهترین نتیجه را داشت، همینطور سیگما ۱ هم نتیجه خوبی رو در بر داشت. بدین ترتیب تابع Gaussian_derivative_filter را پیاده سازی کردم، که باگرفتن سایز و سیگما یک فیلتر با سایز مورد نظر از مشتق گوسی می سازد.
با پرینت کردن این فیلترها ، که با اجرای کد هم پرینت میشود، آن ها را نمایش می دهم:

```
gaussian derivative filter in x-axis:
[[ 5.89238410e-05  4.78559558e-04  1.07237757e-03  0.00000000e+00
  -1.07237757e-03 -4.78559558e-04 -5.89238410e-05]
 [ 7.17839338e-04  5.83004893e-03  1.30642333e-02  0.00000000e+00
  -1.30642333e-02 -5.83004893e-03 -7.17839338e-04]
 [ 3.21713271e-03  2.61284666e-02  5.85498315e-02  0.00000000e+00
  -5.85498315e-02 -2.61284666e-02 -3.21713271e-03]
 [ 5.30415514e-03  4.30785586e-02  9.65323526e-02  0.00000000e+00
  -9.65323526e-02 -4.30785586e-02 -5.30415514e-03]
 [ 3.21713271e-03  2.61284666e-02  5.85498315e-02  0.00000000e+00
  -5.85498315e-02 -2.61284666e-02 -3.21713271e-03]
 [ 7.17839338e-04  5.83004893e-03  1.30642333e-02  0.00000000e+00
  -1.30642333e-02 -5.83004893e-03 -7.17839338e-04]
 [ 5.89238410e-05  4.78559558e-04  1.07237757e-03  0.00000000e+00
  -1.07237757e-03 -4.78559558e-04 -5.89238410e-05]]
```

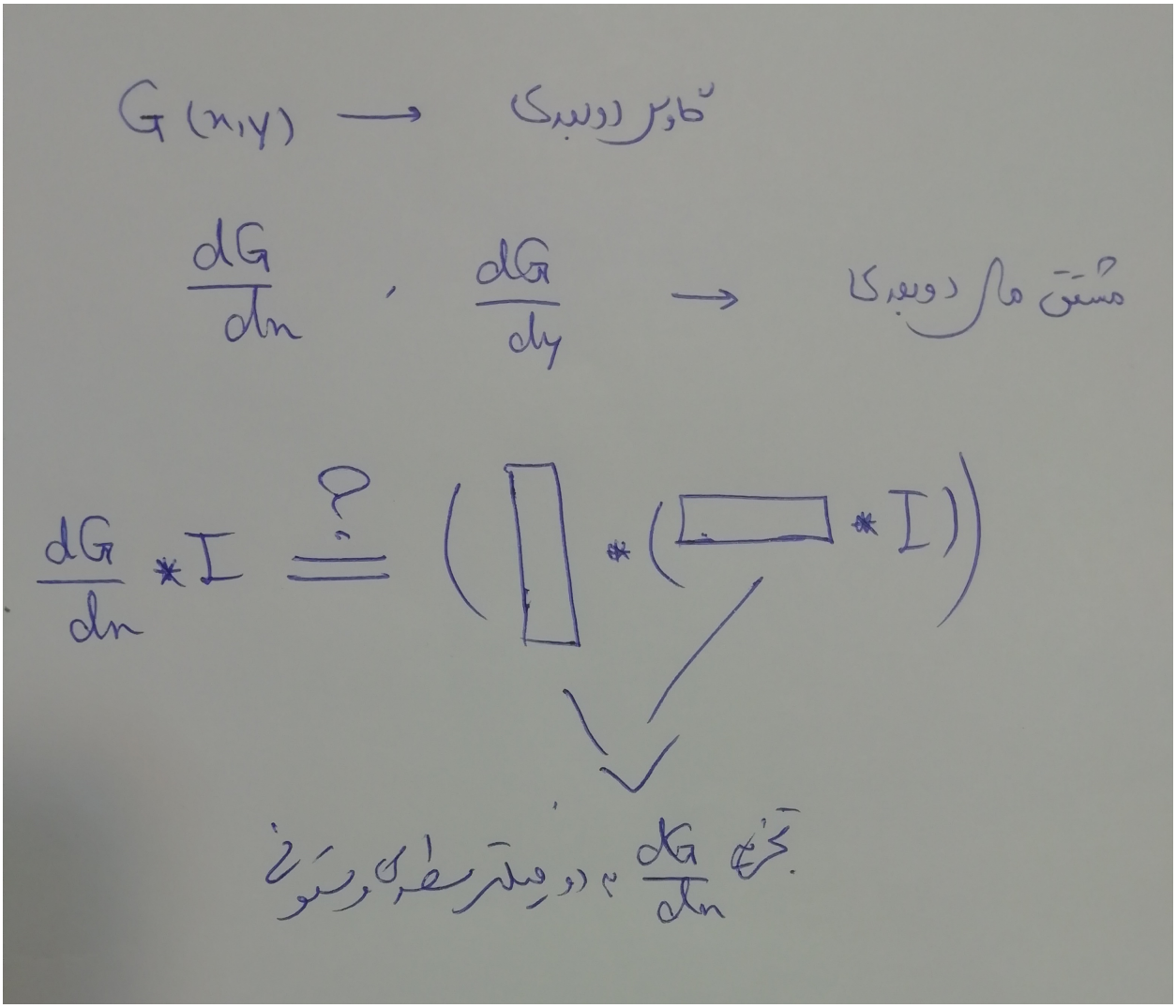
```
gaussian derivative filter in y-axis:
[[ 5.89238410e-05  7.17839338e-04  3.21713271e-03  5.30415514e-03
   3.21713271e-03  7.17839338e-04  5.89238410e-05]
 [ 4.78559558e-04  5.83004893e-03  2.61284666e-02  4.30785586e-02
   2.61284666e-02  5.83004893e-03  4.78559558e-04]
 [ 1.07237757e-03  1.30642333e-02  5.85498315e-02  9.65323526e-02
   5.85498315e-02  1.30642333e-02  1.07237757e-03]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [-1.07237757e-03 -1.30642333e-02 -5.85498315e-02 -9.65323526e-02
  -5.85498315e-02 -1.30642333e-02 -1.07237757e-03]
 [-4.78559558e-04 -5.83004893e-03 -2.61284666e-02 -4.30785586e-02
  -2.61284666e-02 -5.83004893e-03 -4.78559558e-04]
 [-5.89238410e-05 -7.17839338e-04 -3.21713271e-03 -5.30415514e-03
  -3.21713271e-03 -7.17839338e-04 -5.89238410e-05]]
```

همانطور که در کلاس گفته شد فیلتر مشتق گاوس جدایی پذیر است زیرا رنک ماتریس آن ۱ است یا به عبارت دیگر تعداد سطرهای مستقل خطی اش ۱ باشد. از طرف دیگر با نگاه به معادله آن هم به وضوح مشخص است که می توان آن را به صورت ضرب دو ماتریس که یکی فقط x و دیگری فقط y دارد نوشت. همین کار را برای آن انجام می دهیم و بدین صورت به ماتریس هایی سطری و ستونی آن را تبدیل میکنیم. بدین منظور روابط ریاضی آن را نوشتم و آن ها را تابع هایی کردم که در helper.py آمده است. سپس برای هرکدام از این ۴ تابع(که فرمول های مشتق گوسی تجزیه شده است) ۴ تابع دیگر نوشتم که فیلترها را بسازند. این تابع ها با گرفتن سایز و سیگما فیلتر(ماتریس) مورد نظر را ساخته و خروجی میدهند. تابع هایی که فرمول ها را میدهند با نام هایی row_derivative_in_x_axis همچون این هستند و تابع هایی که فیلتر ها را میسازند نامی همچون row_gaussian_derivative_in_axis_filter هستند. فیلتر هایی که ساخته می شوند هم در روند اجرای کد پرینت می شوند که در اینجا هم آن ها را نشان می دهم:

```
Row gaussian derivative filter in x-axis:
[[ 0.01329555  0.10798193  0.24197072  0.         -0.24197072 -0.10798193
  -0.01329555]]
==========================================
Col gaussian derivative filter in x-axis:
[[0.00443185]
 [0.05399097]
 [0.24197072]
 [0.39894228]
 [0.24197072]
 [0.05399097]
 [0.00443185]]
```

```
Row gaussian derivative filter in y-axis:
[[0.00443185 0.05399097 0.24197072 0.39894228 0.24197072 0.05399097
  0.00443185]]
==========================================
Col gaussian derivative filter in y-axis:
[[ 0.01329555]
 [ 0.10798193]
 [ 0.24197072]
 [ 0.        ]
 [-0.24197072]
 [-0.10798193]
 [-0.01329555]]
```

حال برای بدست آوردن مشتق تصویر در دو راستای افقی و عمودی با استفاده از فیلتر های تجزیه شده مطابق صورت سوال عمل میکنم: بدین صورت که ابتدا فیلتر سطری در جهت x و y را در تصویر اصلی کانوالو میکنم. سپس فیلترهای ستونی تجزیه شده را در تصاویر حاصل از مرحله قبل کانوالو میکنم. بدین ترتیب تصاویر حاصله که hor_col و ver_col هستند بدست می آیند.
این تصاویر در واقع مشتق تصویر نسبت به x و مشتق تصویر نسبت به y هستند.
حال اینبار از فیلتر های دوبعدی استفاده میکنیم و با کانوالو کردن فیلترهای مشتق گاوسی دوبعدی که در ابتدا بدست آورده بودیم در تصویر اصلی به ریزالت های x_2d و y_2d می رسیم.
برای آنکه نشان دهیم که این دو کار معادل هم هستند یا به عبارت دیگر برای اثبات این قضیه:



باید ماتریس های حاصله را مقایسه کرد. این کار در کد انجام شده است(یک تابع check_equality در helper.py نوشته شده است) که با استفاده از آن این دوماتریس را مقایسه میکنیم که مشاهده میشود با دقت 0.01 این دو ماتریس باهم برابر هستند.

```
==================================================
x is equal to hor_col
y is equal to ver_col
==================================================
```

حال با استفاده از تابع np.hypot گرادیان عکس ها را محاسبه میکنیم.

سپس با استفاده از تابع np.arctan2 جهت گرادیان تصویر را بدست می آوریم که با پخش کردن آن در بازه ۲۵۵-۰ آن را ذخیره میکنیم.

پس از این با استفاده از threshold پیکسل هایی که کمتر از ۲۰ بودند را حذف کردم. نتیجه بد نبود اما مشکلی که وجود داشت وجود پیکسل های متراکم در عکس بود. به عبارت دیگر خط ها خیلی پر رنگ بودند. برای درست کردن این موضوع از تکنیکی که در کتاب بود استفاده کردم و تابع clean را پیاده سازی کردم. این تابع، ماتریس های grad و theta رو دریافت میکنه و براساس مقدار تتا که در چه زاویه ای هست، تصمیم میگیره که در grad به ازای هر پیکسل اگه در یک جهت (که این جهت به وسیله مقدار تتا مشخص میشود) یک پیکسل دیگه وجود داره که مقدارش از پیکسل فعلی بیشتره، این پیکسل رو صفر میکنه. (این تکنیکی است که در متن کتاب آمده است) پس از انجام این کار، عکس را پخش میکنیم(۲۵۵-۰) سپس پیکسل های زیر ۲۰ را حذف و بالاتر از ۲۰ ها را به ۲۵۵ مپ میکنیم. بدین ترتیب به result نهایی رسیده ایم.