تمرین سری ۲ تمرین شماره ۴

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.

در این فایل، اقدام به ساختن یک کلاس به اسم Image کردم تا کار با image ها و اعمال کارهای مختلف روی آن راحت تر شود. بدین ترتیب که درابتدا یک image (یا به عبارت دیگر یک np.array) به عنوان ورودی میگیرد. تابع های get_image و set_image برای آن پیاده سازی شده است که به ترتیب عکس (array) را خروجی می دهد و یک عکس (array) را به عنوان image اصلی instance ست میکند. تابع save که ایمیج را ذخیره میکند. تابع save_distributed که ابتدا عکس را پخش میکند(۲۵۵-۰) وسپس آن را ذخیره می کند. همچینی تابع هایی برای گرفتن عرض و ارتفاع و چنل های مختلف عکس پیاده سازی شده است . تابعی تحت عنوان mix_channels که یک لیست از چنل ها میگیرد و یک عکس RGB خروجی میدهد. همینطور توابعی برای کار کردن با fft هم در آن پیاده سازی شده است که در حین توضیح روند کار به آنها اشاره خواهم کرد.

توضیح روند اصلی کد:

ابتدا عکس های نزدیک و دور خوانده میشوند و سپس یک instance از کلاس Image برای آنها میسازم.

حال باید این دو عکس رو جوری تغییر بدیم که چشم های دو شخصیت رو هم بیفته. بدین منظوری یک تابع به نام fix_images در فایل helper.py پیاده سازی شده است که از آن استفاده میکنم و به عکس های فیکس شده می رسم و آن ها را ذخیره میکنم.

برای نمایش لگاریتم عکس ها در حوزه فرکانس،از تابع get_fft_log_image که یکی از تابع های کلاس Image است و در helper.py پیاده سازی شده است استفاده می کنم. این تابع از تابع get_fft_log_channels تک تک چنل ها را که لاگ آنها در حوزه فرکانس گرفته شده است استفاده میکند و باترکیب کردن آنها با استفاده از تابع mix_channels عکس نهایی را خروجی میدهد. تابع get_fft_log_channels هم بدین صورت عمل میکند که عکس هایی که توسط تابع get_shifted_fft_channels روی هر چنل آن عمل fft و سپس fftshift انجام شده است را دریافت میکند وسپس بعد از انجام np.abs و np.log روی هرچنل، لیست چنل هارا خروجی میدهد.

پس روند انجام کار بدین صورت بود:

get_shifted_fft_channels:

لیست چنل های عکس را میگیرد. روی هر چنل تابع های fftو سپس fftshift را اعمال میکند و لیست چنل های تغییرداده شده را خروجی میدهد.

get_fft_log_channels:

لیست چنل های شیفت داده شده(خروجی تابع بالا) را میگیرد و بر روی هر چنل آن عمل np.abs وسپس np.log را انجام می دهد و لیست چنل های تغییر داده شده را خروجی میدهد.

get_fft_log_image:

لیست چنل های بالا را میگیرد(خروجی fft_log_channels) وبا استفاده از تابع mix_channels آن هارا ترکیب کرده و یک عکس RGB خروجی می دهد.

بدین ترتیب لاگ تصویر های near و far را بدست آوردیم و ذخیره کردیم.

حال باید فیلترهای lowpass و highpass را بسازیم. در صورت سوال گفته شده که از تابع گاوس دو بعدی با انحراف معیار s استفاده کنید. اما استفاده از این تابع نتایج خوبی به همراه نداشت. در نتیجه تصمیم گرفتم که از تابعی که در اسلایدها تحت عنوان Gaussian lowpass filter استفاده کنم که معادله آن بدین صورت است:

Gaussian Lowpass Filter

$$H(u,v) = e^{-D^2(u,v)/2D_0^2}$$

این تابع را در helper.py و تابعی به اسم Gaussian_lowpass پیاده سازی کرده ام. سپس تابعی به اسم make_lowpass_filter نوشتم که یک ماتریس(فیلتر) با استفاده از تابع بالا می سازد و خروجی می دهد.
بدین ترتیب تابع های lowpass و highpass را ساختم و خروجی را ذخیره کردم:

```
# create lowpass and highpass filters
lowpass_filter = make_lowpass_filter(width=images_width, height=images_height, D0=20)
distributed_lowpass_filter = lowpass_filter / lowpass_filter.max() * 255
cv2.imwrite(filename="out/Q4_08_lowpass_20.jpg", img=distributed_lowpass_filter)

highpass_filter = 1 - make_lowpass_filter(width=images_width, height=images_height, D0=20)
distributed_highpass_filter = highpass_filter / highpass_filter.max() * 255
cv2.imwrite(filename="out/Q4_07_highpass_20.jpg", img=distributed_highpass_filter)
```

برای cutoff هم تابعی تحت عنوان cutoff در فایل helper.py نوشتم که یک فیلتر خروجی میدهد که فقط پیکسل های که فاصله آنها از مرکز کمتر از d0 هست 1 هست و بقیه صفر هستند.

سپس این فیلترهای کات آف را متناسب سایز عکس ها ساختم و آن ها را ضرب درایه به درایه در فیلترهای lowpass و highpass کردم.

```
# cutoff
cutoff_filter_for_lowpass = cutoff(width=images_width, height=images_height, D0=20)
lowpass_cutoff = np.multiply(cutoff_filter_for_lowpass, lowpass_filter)
distributed_lowpass_cutoff = lowpass_cutoff / lowpass_cutoff.max() * 255
cv2.imwrite(filename="out/Q4_10_lowpass_cutoff.jpg", img=distributed_lowpass_cutoff)

cutoff_filter_for_highpass = 1 - cutoff(width=images_width, height=images_height, D0=15)
highpass_cutoff = np.multiply(cutoff_filter_for_highpass, highpass_filter)
distributed_highpass_cutoff = highpass_cutoff / highpass_cutoff.max() * 255
cv2.imwrite(filename="out/Q4_09_highpass_cutoff.jpg", img=distributed_highpass_cutoff)
```

حال بایستی فیلتر های بدست آمده را درتصاویرمتناظرشان اعمال کنم :

برای این موضوع تابع get_image_of_applied_cutoff_filter_in_frequency_domain را نوشتم. این تابع یک cutoff فیلتر میگیرد و باکمک تابع apply_cutoff_filter_on_channels_in_frequency_domain این فیلتر را روی هرکدام از چنل های آن به صورت جدا اجرا میکنم و سپس خروجی حاصل(یک لیست از چنل هایی که این cutoff روی هرکدام از چنل های آن اجرا شده است) را با کمک mix_channels به یک عکس RGB تبدیل میکنم.

همچینین با اعمال این فیلترها و بعد از آن بردن آن به حوزه مکان میتوان به این موضوع پی برد که فیلترهای بالاگذر چگونه برای پیداکردن edge و نویز در تصویر خوب هستند و فیلتر های پایین گذر چگونه برای smooth کردن عکس به کار میروند. من این دو را هم خروجی گرفتم با اینکه در صورت سوال نیامده بود.

حال بایستی این دو تصویر را با کمک میانگین گیری وزن دار ترکیب کنیم. بدین منظور از تابع mix_near_and_far_frequency_domain استفاده میکنیم. این تابع چنل های عکس ها بعد از اعمال cutoff روی آن را از ما میگیرد همراه با یک ضریب آلفا و بتا که مشخص میکند تاثیر کدام عکس بیشتر باشد(نزدیک یا دور).
این تابع دو خروجی دارد. یکی mix_channels و دیگری mix_frequency . mix_channels لیستی از چنل هاست که که هرکدام از اعضای آن یک چنل هستند که با میانگین گیری وزن دار ترکیب شده اند. Mix_freqeuncy هم ترکیب این چنل هاست که یک عکس RGB است.

بدین ترتیب فقط مانده است خروجی عکس در حوزه مکان. که کافی است mix_channels خروجی تابع بالا را گرفته و هرکدام از چنل های آن را جداگانه به حوزه مکان برده وسپس با ترکیب آنها عکس هیبریدی حاصل را خروجی بگیریم. این کار را تابع mix_near_and_far_spatial_domain انجام میدهد.
در نهایت هم یک عکس کوچکتر ذخیره میکنم.

سعی کردم تا جایی که ممکن است تابع هایم را کوچک تر کنم تا تمیزی کد بیشتر شود. اما اگر کمی گیج کننده شده است معذرت می خواهم :) برای هر تابع هم سعی کردم توضیح بذارم تا مشکلی در خواندن کد ها وجود نداشته باشد.
باتشکر :)