

به نام خدا



اصول پردازش تصویر

تمرین سری دوم

استاد درس

دکتر مصطفی کمالی تبریزی

نیمسال اول ۱۴۰۱-۱۴۰۰

نکات تکمیلی :

۱. برای ران کردن کد ها ، نیاز به لایبرری های Scipy , OpenCV , Numpy و Matplotlib داریم.
۲. عکس های ورودی را در پوشه resources قرار دهید.
۳. بعد از اجرای هر کد ، نتایج آن در پوشه Result قرار میگیرد. نتایج اصلی در همین پوشه و در صورتی که نتایج دیگری نیز ایجاد شده باشد، در زیرپوشه های دیگری به تفکیک هر تمرین قرار گرفته است.
۴. در لینک زیر نیز میتوانید کلیه نتایج حاصل را مشاهده کنید :

<https://mega.nz/folder/n9ggVaDD#ni4WDDdBO0xyHkDZ0EwX7g>

۵. نتایج res01 تا res31 در پوشه اصلی لینک بالا قرار گرفته اند. در صورتی که برای تمرینی، نتایج و خروجی های دیگری نیز درست شده باشد، آنها را به تفکیک هر تمرین در زیر پوشه ها قرار داده ام.
۶. در اکثر تمرین ها، برای آنکه نتیجه محاسبات دقیقتر باشد، فرمت عکس ورودی (که uint8 هست) را تبدیل به float میکنم.

مشورت های انجام شده :

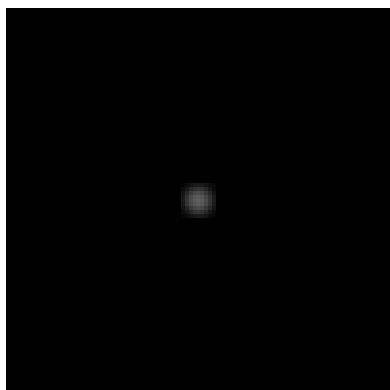
سوال ۲:

۱. محمدمهدی زارع (ایده ی کراپ کردن patch)
۲. علی شفیعی (ایده ی کراپ کردن و اسکیل کردن patch)

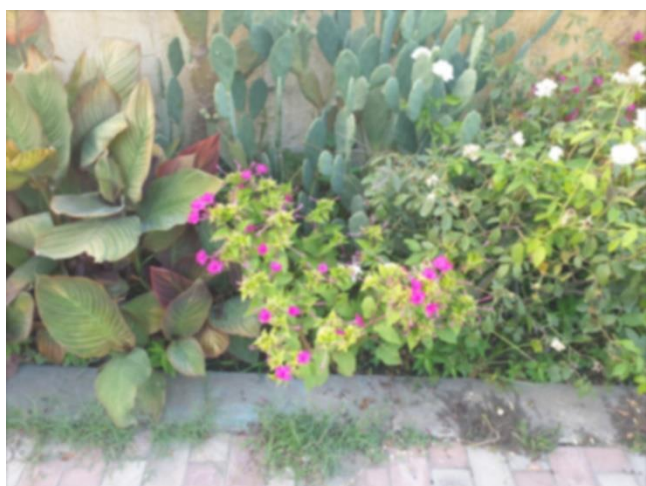
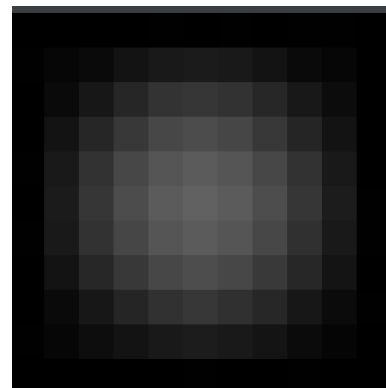
سوال اول

الف

بعد از لود کردن عکس ، با استفاده از تابع `cv2.GaussianBlur()` ، یک فیلتر گاوسی با اندازه 9×9 (و انحراف معیار ۲ در راستای x , y) بر روی اعمال میکنیم تا $f * g$ را تشکیل دهیم. (این عکس را `blurred_image` مینامیم. همچنین برای نمایش فیلتر گاوسی، آن را در یک عکس 101×101 پیکسل نمایش میدهیم تا معلوم باشد. سپس عکس اولیه را از این `blurred_image` کم میکنیم تا `unsharp_mask` بدست آید.) هنگام نمایش این ماسک ، آن را طوری اسکیل میکنیم که در بازه ی $[0,255]$ قرار بگیرد تا مقادیر منفی آن از بین بروند و قابل نمایش شود. در نهایت با ضرب $k = 2$ ، این `unsharp_mask` را به عکس اولیه اضافه میکنیم. تصاویر `res1` , `res2` , `res3` , `res4` به ترتیب زیر هستند :



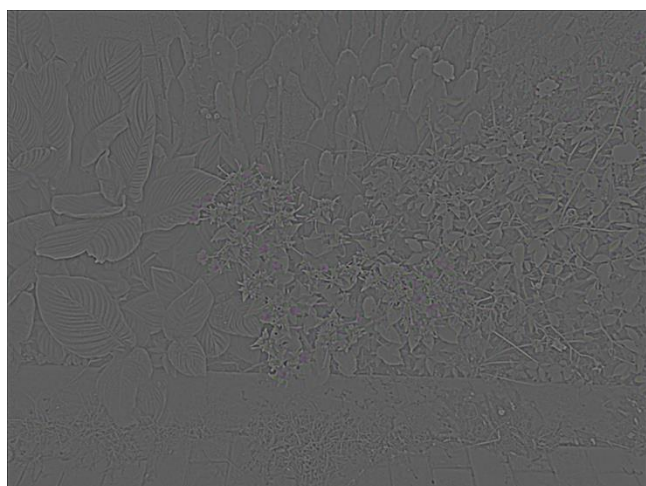
نسخه زوم شده فیلتر گاوسی



نسخه بلور شده



نسخه اصلی



`unsharp_mask`



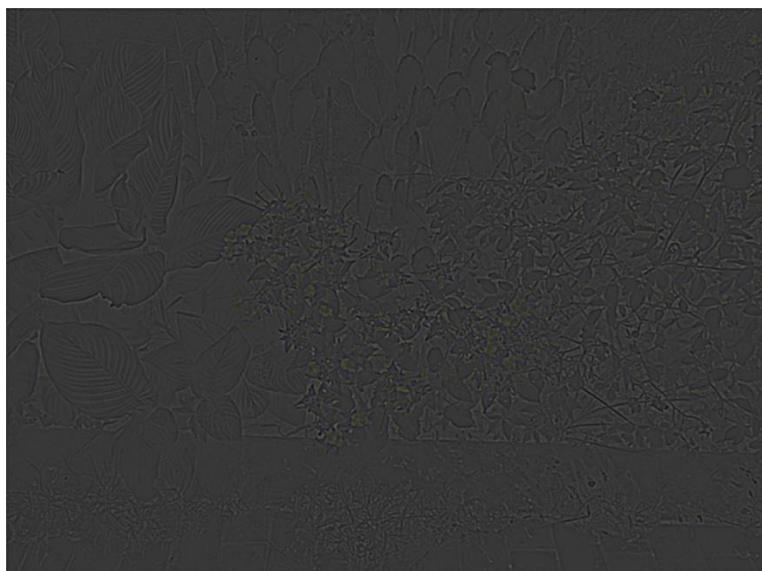
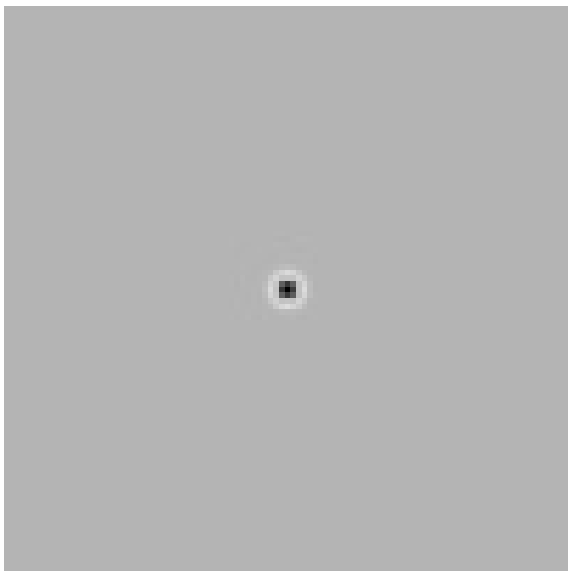
نسخه نهایی

(ب)

ابتدا به ازای سه کانال r , g , b ، با استفاده از تابع `scipy.ndimage.filters.gaussian_laplace` ، تابع LoG آن ها را حساب میکنیم. سیگمای مورد استفاده این تابع در هر دو جهت x , y برابر ۱ است.

سپس این ماتریس حاصل (unsharp mask زیر) را با ضریب $k=3$ از عکس اصلی کم میکنیم تا نتیجه مطلوب حاصل شود .

برای نمایش فیلتر LoG نیز، یک ماتریس 101×101 تمام صفر میسازیم که فقط درایه وسط آن سفید است و سپس تابع بالا را روی آن صدا میزنیم.



تابع create_mask : با استفاده از این تابع ، میتوانیم یک تابع گاوسی low_pass یا high_pass هم اندازه عکس اصلی، بسازیم.

Gaussian

$$H(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$$

که در آن D_0 یک مقدار CUT_OFF در حوزه فرکانس خواهد بود(و همانند انحراف معیار در حوزه مکان عمل خواهد کرد). برای این قسمت از $D_0 = 130$ استفاده میکنیم.

تابع fourier_c : ابتدا تبدیل فوریه عکس اصلی را میگیریم و آن را شیفต์ میدهیم. مقدار abs این تبدیل فوریه را ذخیره میکنیم تا بعد آن را نمایش دهیم. سپس یک ماسک high_pass میسازیم. و آن را در تبدیل فوریه خود ضرب میکنیم.

حال داریم :

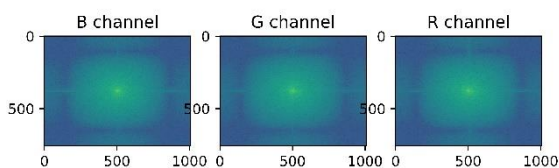
$$\mathcal{F}^{-1}\{(1 + kH_{HP}).F\} = \mathcal{F}^{-1}\{(F + kH_{HP}.F)\}$$

بنابراین ، حاصلضرب تبدیل فوریه و ماسک hp را با یک ضریب $k=7$ به تبدیل فوریه عکس اضافه میکنیم.

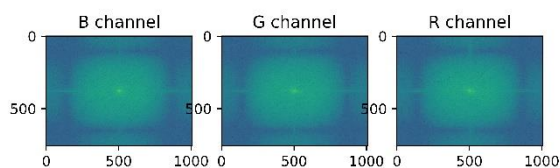
مجددا مقدار abs آن را حساب میکنیم و نگه میداریم. در نهایت نیز، یک شیفต์ وارون و سپس وارون فوریه میگیریم تا به حوزه مکان برگردیم. مقدار real وارون فوریه را حساب میکنیم و همراه دو مقدار abs که از قبل حساب کرده بودیم، خروجی میدهیم.

برنامه اصلی : به ازای سه کانال رنگی r , g , b ، تابع fourier_c را صدا میزنیم. و نتایج بدست آمده را با هم ترکیب کرده و خروجی میدهیم.

magnitude of original image



magnitude of $(1+kH).F$



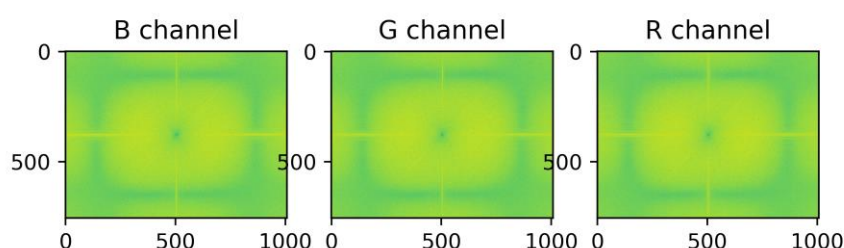
$$H(u,v) = -4\pi^2 \left[(u - P/2)^2 + (v - Q/2)^2 \right] \\ = -4\pi^2 D^2(u,v)$$

تابع create_mask: همانند قسمت قبل، با استفاده از این تابع، یک تابع لاپلاسی در حوزه فرکانس میسازیم. (برای ساخت، از فرمول روبرو استفاده میکنیم).

تابع fourier_d: تبدیل فوریه عکس را میگیریم و آن را شیف می‌دهیم. سپس آن را در ماسک بالا ضرب میکنیم. مقدار abs آن را حساب کرده و نگه میداریم. در ادامه شیف معکوس می‌دهیم و در نهایت وارون فوریه میگیریم و مقدار real آن را به همراه abs که بدست آورده بودیم خروجی می‌دهیم.

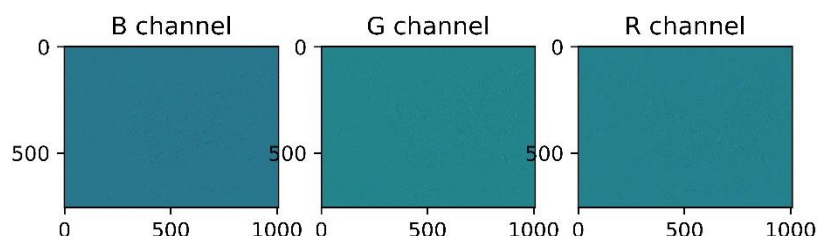
برنامه اصلی: ابتدا عکس را به ۲۵۵ تقسیم میکنیم تا به دامنه ی (0,1) برود. سپس به ازای سه کانال رنگی r, g, b، تابع fourier_d را صدا میزنیم. سپس نتایج abs بدست آمده را با یکدیگر ترکیب کرده و خروجی می‌دهیم:

magnitude of $4.\pi^2.(u^2 + v^2)F$



مقادیر real بدست آمده را نیز با یکدیگر ترکیب میکنیم. این مقادیر به صورت زیر هستند (باید زوم کنید تا مشخص شوند)

magnitude of $F^{-1} \{ 4.\pi^2.(u^2 + v^2)F \}$




حال این مقادیر real را نیز نرمال میکنیم. (هر کدام را به ماکسیمم همان ماتریس تقسیم میکنیم. به این صورت هر کدام به بازه ی (-1,1) میروند.) و در نهایت با ضریب $k=-2$ ، با عکس اصلی جمع میکنیم تا تصویر نهایی حاصل شود:



نتایج نهایی :

2	انحراف معیار فیلتر گاوس	قسمت الف
2	مقدار a	
1	انحراف معیار فیلتر گاوس	قسمت ب
3	مقدار k	
7	مقدار k	قسمت ج
130	مقدار D_0	
-2	مقدار k	قسمت د

4. Normalized Cross-Correlation

- Goal: find  in image
- Method 4: Normalized Cross-Correlation

از روش ncc استفاده میکنیم :

$$h[m, n] = \frac{\sum_{k,l} (g[k, l] - \bar{g})(f[m + k, n + l] - \bar{f}_{m,n})}{\left(\sum_{k,l} (g[k, l] - \bar{g})^2 \sum_{k,l} (f[m + k, n + l] - \bar{f}_{m,n})^2 \right)^{0.5}}$$

mean template
mean image patch

ابتدا در نظر بگیرید که :

$$g'[k, l] = g[k, l] - \bar{g}$$

حال تابع بالا را به صورت دیگری بنویسیم :

$$h[m, n] = \frac{\sum_{k,l} (g'[k, l] * f[m + k, n + l]) - \bar{f}_{m,n} \sum_{k,l} g'[k, l]}{\sqrt{(\sum_{k,l} (g'[k, l])^2) * \left(\sum_{k,l} (f[m + k, n + l])^2 - 2\bar{f}_{m,n} \sum_{k,l} f[m + k, n + l] + k * l * \bar{f}_{m,n}^2 \right)}}$$

که میتوان به این صورت، ماتریس H را به صورت چند عملیات ماتریسی به سرعت محاسبه کرد.

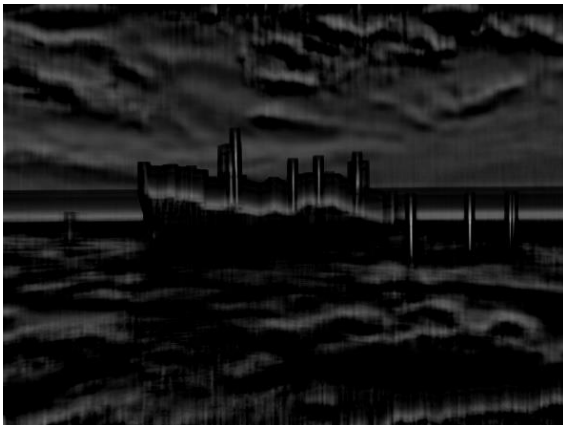
۱- $\sum_{k,l} (g'[k, l] * f[m + k, n + l])$: که این عبارت، ماتریس حاصل از کانولوشن g' و f است.

۲- $\bar{f}_{m,n}$: این ماتریس را میتوان از حاصل کانولوشن f و یک ماتریس تمام یک به اندازه patch و سپس تقسیم آن بر مجموع تعداد درایه های patch بدست آورد.

۳- $\sum_{k,l} g'[k, l]$: که این عبارت به راحتی قابل محاسبه است. (یک عدد ثابت است).

۴- $\sqrt{(\sum_{k,l} (g'[k, l])^2)}$: این عبارت نیز یک عدد ثابت است و به راحتی قابل محاسبه است.

۵- $\sum_{k,l} (f[m + k, n + l])^2$: کافی است ابتدا ماتریس f را به توان دو برسانیم و بعد حاصل کانولوشن f^2 و یک ماتریس تمام یک به اندازه patch خواهد بود.



الگوریتم :

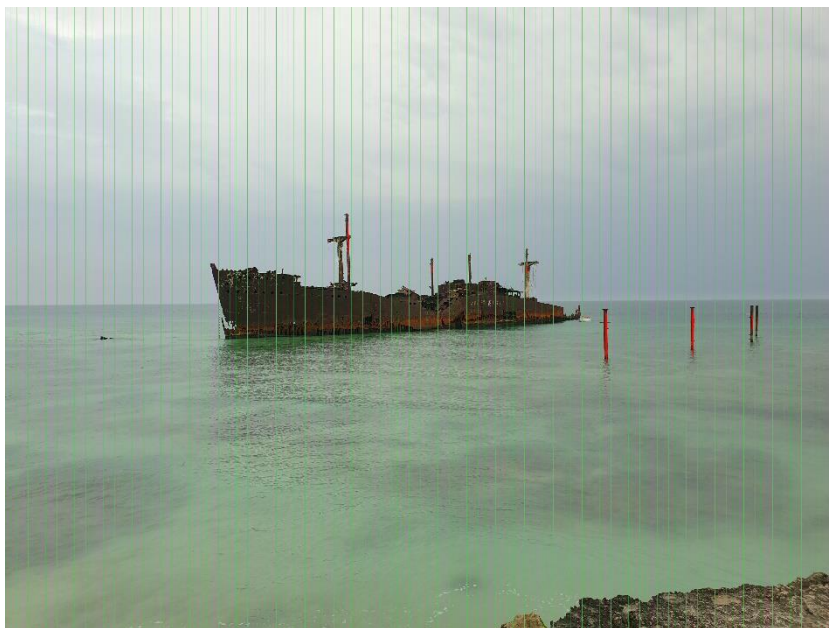
۱. ابتدا عکس کشتی را لود میکنیم و آن را تبدیل به یک عکس خاکستری میکنیم. عکس patch را نیز لود میکنیم، آن را تبدیل به یک عکس خاکستری میکنیم، از هر چهار طرف آن، ۵۰ ردیف یا ستون حذف میکنیم و نتیجه بدست آمده را تا ۶۰ درصد کوچک میکنیم.

۲. سپس با استفاده از روابط ماتریسی بالا، تابع $corr$ را مینویسیم و حاصل $corr(img, patch)$ را بدست می آوریم. (عکس روبرو بالا)

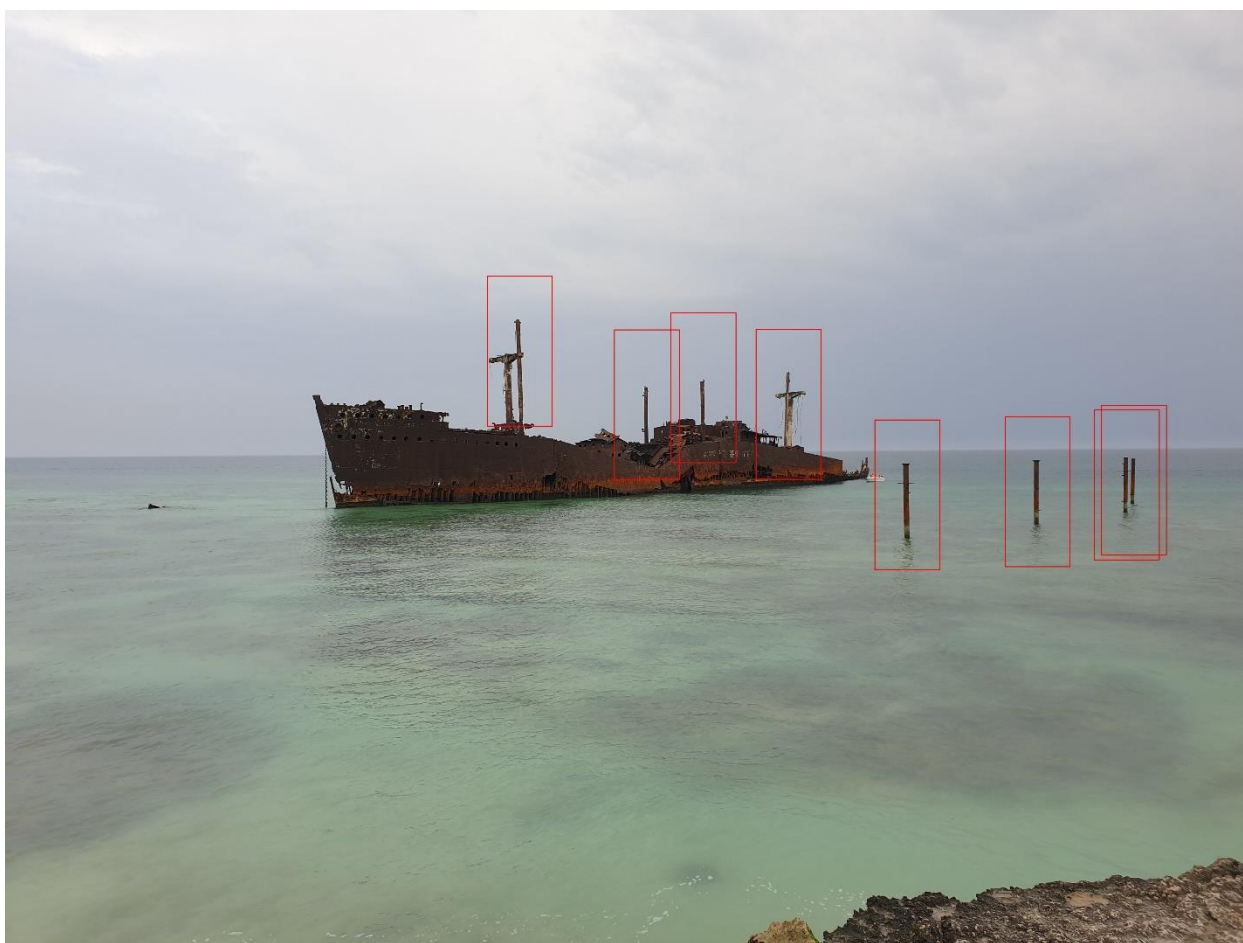
۳. یک $threshold = 0.45$ در نظر میگیریم و تمام پیکسل هایی که در عکس روبرو از این مقدار بیشتر باشند را در نظر میگیریم. (نتیجه شبیه عکس روبرو پایین خواهد بود)



۴. عکس را به صورت یک سری نوار عمودی در نظر میگیریم.



۵. در هر کدام از این نوار ها، ماکسیمم از بین تمام پیکسل هایی که که شرایط *threshold* را برآورده کرده بودند را پیدا میکنیم و به مرکز این پیکسل یک مستطیل رسم میکنیم.



سوال سوم)

نقاط چهارگوشه ی هرکدام، به صورت دستی انتخاب شده و هارد کد شده اند. (این نقاط به ترتیب بالا چپ، پایین چپ، پایین راست و بالا راست هر کتاب هستند)

با استفاده از تابع projective، ابتدا با دریافت نقاط مذکور، طول و عرض آن کتاب را ذکر میکنیم و یک عکس جدید با این ابعاد میسازیم. نقاط پایانی پراجکشن را نیز، چهار نقطه ی گوشه ی این عکس جدید در نظر میگیریم.

حال یک inverse wrapping انجام میدهیم، یعنی فرض میکنیم که میخوایم عکس دوم را تبدیل به عکس اصلی کنیم.

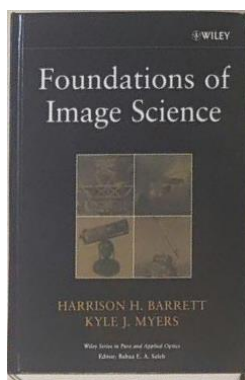
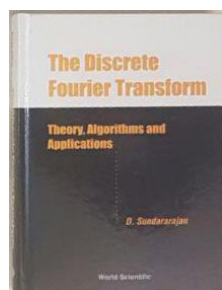
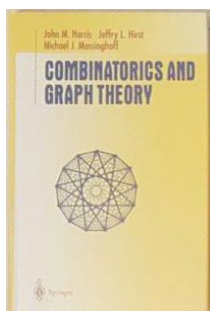
در این صورت، با استفاده از تابع cv2.findHomography، ماتریس تبدیل مورد نظر (از عکس دوم به عکس اصلی) را پیدا میکنیم. (ماتریس h)

سپس به ازای هر پیکسل (x, y) از عکس دوم، یک بردار به صورت $\text{temp} = [x, y, 1]^T$ در نظر میگیریم و حاصل ضرب ماتریس $h * \text{temp}$ را حساب میکنیم. و سپس بردار حاصل را بر درایه سوم آن تقسیم میکنیم تا نرمال شود و یک x' , y' بدست بیاید.

$$\begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad \begin{aligned} x' &= \frac{u}{w} \\ y' &= \frac{v}{w} \end{aligned}$$

این x' , y' مختصات یک پیکسل در عکس اصلی است. حال یک همسایگی چهار پیکسلی از این مختصات x' , y' در نظر میگیریم و با انجام یک درون یابی خطی توسط تابع interpolate، یک مقدار مناسب مانند f بدست می آوریم. و در نهایت این مقدار f را در پیکسل (x,y) عکس دوم قرار میدهیم. (اگر x' , y' یک مختصات غیر معتبر (مثلا منفی) داشته باشند، آن را در نظر نمیگیریم)

در نهایت نیز عکس را ذخیره میکنیم.



عکس هایی نهایی به این صورت خواهند بود :

```
304
204
[[-9.22369746e-01 -3.47142637e-01 6.68275147e+02]
 [-3.41315896e-01 9.43324559e-01 2.07399749e+02]
 [ 3.96287282e-05 -3.11808158e-05 1.00000000e+00]]
*****
283
214
[[ 1.95372352e-01 -1.02343876e+00 3.64790544e+02]
 [-9.57130844e-01 -2.65778036e-01 7.45146215e+02]
 [ 5.38627687e-05 -1.56945604e-04 1.00000000e+00]]
*****
372
241
[[-4.70012782e-01 -9.01677781e-01 8.17433566e+02]
 [-7.60737966e-01 5.20850884e-01 9.68313288e+02]
 [ 1.18956263e-04 -4.31284629e-05 1.00000000e+00]]
*****
```

ماتریس حاصل برای هر کدام از کتاب های بالا (به ترتیب از چپ به راست) به صورت روبرو است. (دو عدد اول به ترتیب طول و عرض کتاب هر کتاب هستند.)

سوال چهارم



تصاویر انتخاب شده برای این تمرین، تصاویر شرک و پاندای کنگفوکار است :

مطابق صحبت هایی که سر کلاس مطرح شد، این دو عکس در یک محیط دیگر به یکدیگر، بر یکدیگر منطبق شده اند.

میخواهیم کاری کنیم که از نزدیک، عکس شرک، و از دور عکس پاندا نمایان باشد.

بنابراین باید برای شرک از یک ماسک high_pass (که جزئیات آن را حفظ کند) و برای پاندا از یک ماسک low_pass (که کلیات تصویر را حفظ کند) در نظر بگیریم.

Gaussian

برای ساخت ماسک، از فیلتر گاوسی استفاده میکنیم. تابع این فیلتر در دامنه فرکانس به این صورت است :

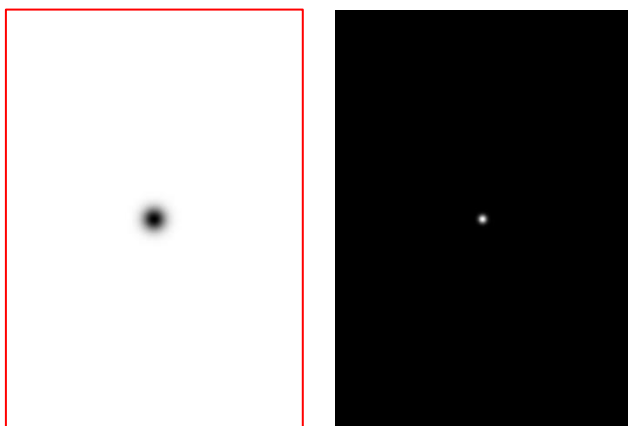
$$H(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$$

که در آن ، D_0 ، عملکردی شبیه به انحراف معیار در حوزه مکان دارد. (در واقع یک frequency cut off در حوزه فرکانس است و میتوان از آن به عنوان یک جور شعاع دایره استفاده کرد).

به هر ترتیب دو ماسک زیر را میسازیم :

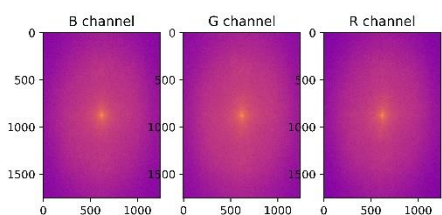
ماسک highpass با $D_0 = 33$

ماسک lowpass با $D_0 = 12$

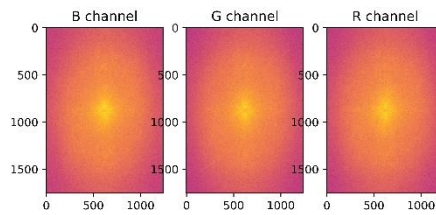


سپس این ماسک ها را در عکس متناظر خود ضرب میکنیم :

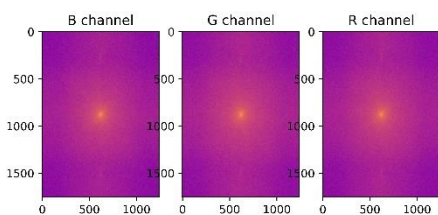
Fourier transform magnitude of near image



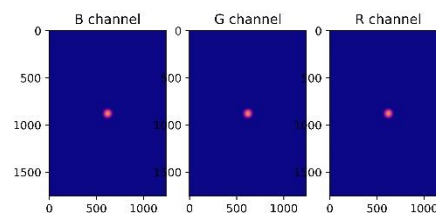
Fourier transform magnitude of Near image with mask



Fourier transform magnitude of Far image

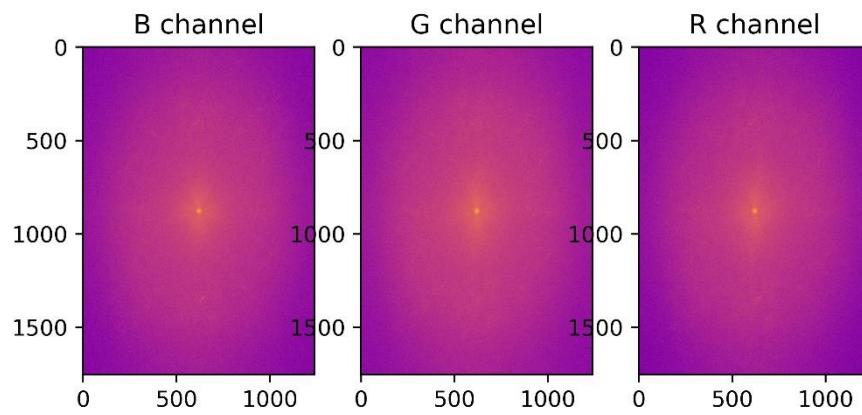


Fourier transform magnitude of Far image with mask



و عکس نهایی در دامنه فرکانس به صورت زیر است :

Fourier transform magnitude of hybrid image



و آنگاه، عکس نهایی در حوزه مکان :

