

به نام خدا



اصول پردازش تصویر

تمرين سري سوم

استاد درس

دکتر مصطفی کمالی تبریزی

نیمسال اول ۱۴۰۰-۱۴۰۱

نکات تكميلی :

۱. برای ران کردن کد ها ، نیاز به لایبریری های Numpy ، OpenCV ، Scipy 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 :

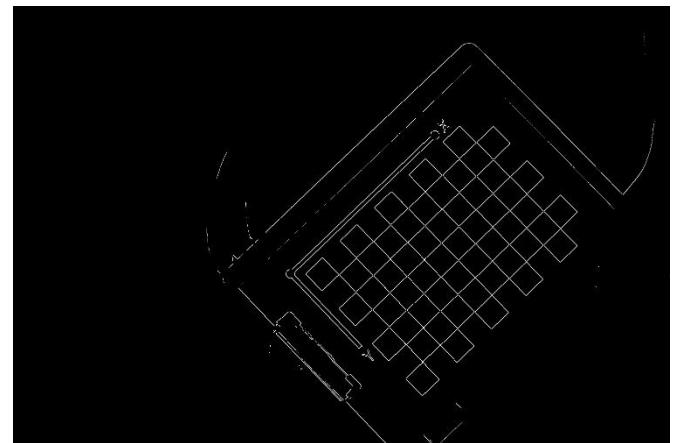
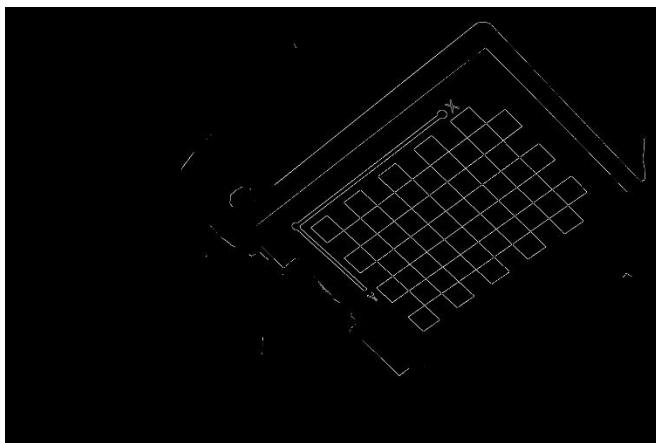
<https://mega.nz/folder/Kp5yxapY#9rbaZ3l-nHmSUELg4DfAQ>

۵. نتایج res01 تا res16 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 انکه نتیجه محاسبات دقیقتر باشد، فرمت عکس ورودی (که uint8 هست) را تبدیل به float میکنم.

سوال اول

مختصاتی که برای عکس در نظر میگیریم، مشابه مختصات دکارتی است: یعنی مبدا را در وسط عکس درنظر میگیریم و همچنین جهت $y +$ را به سمت بالا اختیار میکنیم.(در نتیجه عکس به چهار ناحیه مشابه چهار ربع دستگاه دکارتی تقسیم میشود).

ابتدا با استفاده از تابع cv2.Canny اقدام به لبه یابی عکس میکنیم. حاصل یک ماتریس باینری است که مقادیر غیر لبه 0 و مقادیر لبه دارای مقدار 255 هستند.

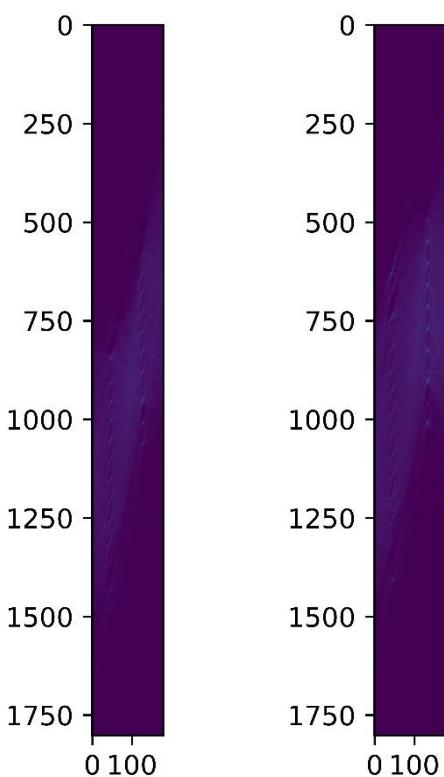


سپس ، فضای هاف خود را تشکیل میدهیم. این فضا یک ماتریس دو بعدی به ابعاد $(2^{*}\text{max_r}) * (\text{num_of_theta})$ است.

$$\text{Num_of_theta} = 180$$

Max_r = بیشترین فاصله از مبدا = نصف قطر عکس.

هر کدام نیز با $\text{step}=1$ گسسته سازی شده اند.



فضای هاف را مطابق مطالب توضیح داده شده در کلاس درست میکنیم. یعنی به ازای تمام پیکسل هایی که روی یک لبه هستند، و تمام مقادیر زاویه از ۰ درجه تا ۱۷۹ درجه، مقدار

$$r = x \cdot \cos(\theta) + y \cdot \sin(\theta)$$

را حساب میکنیم و آنگاه $\text{hough[int(r), \theta]} += 1$ میکنیم.

سپس نوبت به پیدا کردن خطوط میرسد.

با استفاده از تابع **line_finder** این کار را انجام میدهیم.

تعريف میکنیم که در نهایت به دنبال حداکثر ۲۷ خط با قدرت بیشتر از 80° هستیم. (منظور از قدرت، مقدار آن خط در ماتریس هاف است) برای اینکار، هربار، ماکسیمم کلی ماتریس هاف را پیدا میکنیم و آن را به یک لیست اضافه میکنیم. سپس یک همسایگی از آن نقطه را حذف میکنیم و این روند را تا رسیدن به دنبال حداکثر ۲۷ خط با قدرت بیشتر از 80° ادامه میدهیم.(دلیل حذف یک همسایگی این است که خطوطی بسیار نزدیک بهم(که درواقع r و θ بسیار نزدیک بهم دارند و درواقع تقریب های مختلفی از یک خط هستند) را نباید در نظر گرفت، چون به دنبال خطوط متفاوت میگردیم).

سپس خطوط را دسته بندی میکنیم. این دسته بندی را بر حسب زاویه آنها انجام میدهیم و خطوط را به دو دسته تقسیم میکنیم تا بعدا تقاطع آنها را بدست آوریم. سپس، از روی مقادیر r ، θ که در مرحله قبل پیدا کرده بودیم، دو مقدار زیر را محاسبه میکنیم :

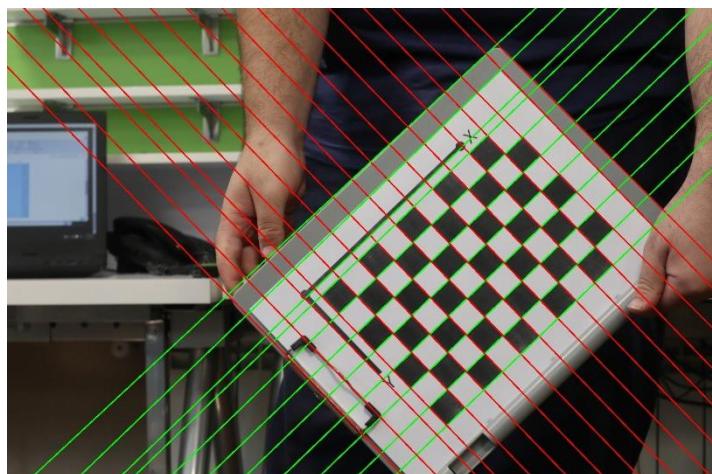
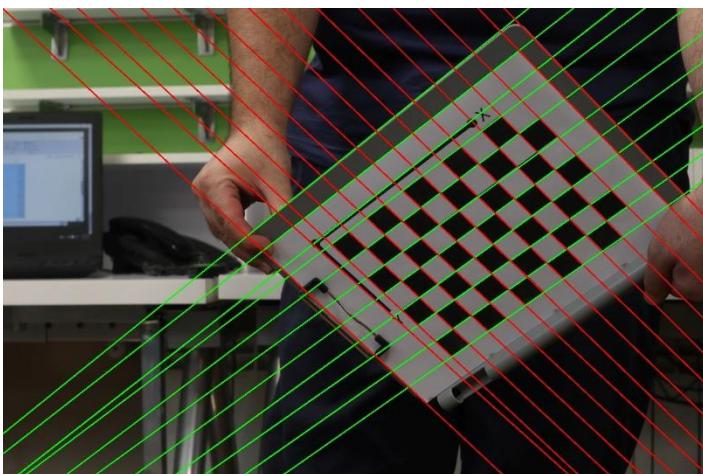
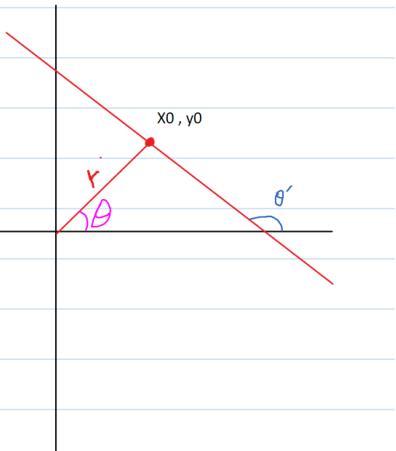
$$x_0 = r \cdot \cos(\theta)$$

$$y_0 = r \cdot \sin(\theta)$$

که مختصات یک نقطه روی خطی است که به دنبال آن هستیم.

آنگاه با استفاده از زاویه θ ، زاویه‌ی خط با راستای افقی (زاویه‌ی θ') را پیدا میکنیم.

به کمک این زاویه شیب خط را بدست می آوریم و در نهایت، دو نقطه دیگر مثل $(x_1, y_1), (x_2, y_2)$ روی آن خط پیدا میکنیم تا به کمک آن بتوانیم خط را رسم کنیم.



بعد از این مرحله ، تصمیم میگیریم که آیا یک خط خوب است یا خیر.

ابتدا طبق رابطه‌ی زیر ([لینک ویکی پدیا](#)) نقطه برخورد دو خط را با توجه به ۲ نقطه روی هر خط پیدا میکنیم :

Given two points on each line [edit]

First we consider the intersection of two lines L_1 and L_2 in 2-dimensional space, with line L_1 being defined by two distinct points (x_1, y_1) and (x_2, y_2) , and line L_2 being defined by two distinct points (x_3, y_3) and (x_4, y_4) .^[1]

The intersection P of line L_1 and L_2 can be defined using determinants.

$$P_x = \frac{\begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix}}$$

$$P_y = \frac{\begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix}}{\begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix}}$$

The determinants can be written out as:

$$(P_x, P_y) = \left(\frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{D}, \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{D} \right)$$

where the denominator is:

$$D = (x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)$$

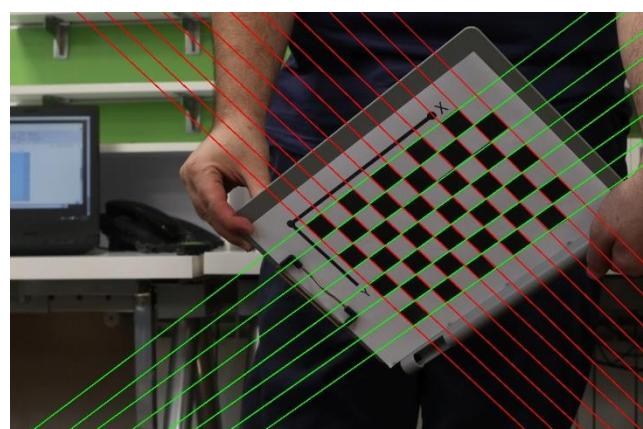
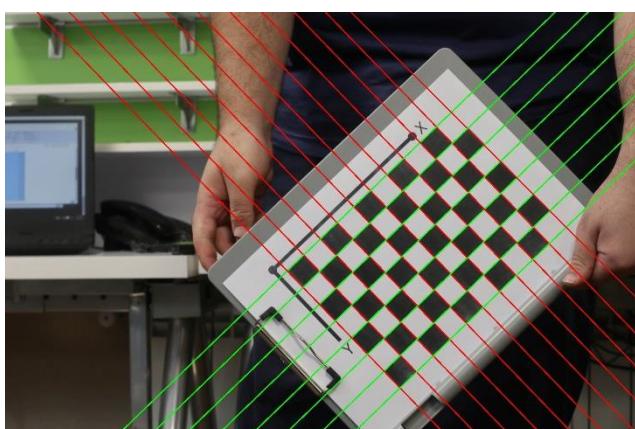
When the two lines are parallel or coincident, the denominator is zero. If the lines are almost parallel, then a computer solution might encounter numeric problems implementing the solution described above: the recognition of this condition might require an approximate test in a practical application. An alternate approach might be to rotate the line segments so that one of them is horizontal, whence the solution of the rotated parametric form of the second line is easily obtained. Careful discussion of the special cases is required (parallel lines/coincident lines, overlapping/non-overlapping intervals).

هنگامی که این نقاط برخورد پیدا شند ، یک همسایگی آن را بررسی میکنیم آیا یک گوشش شطرنجی است یا خیر، برای اینکار ۴ نقطه از چهار طرف آن انتخاب میکنیم و آنها را با یکدیگر مقایسه میکنیم.(اینکار در تابع **is_good** پیاده سازی شده که یک سری عملیات مقایسه ای ساده است که آن را توضیح نمیدهیم.)

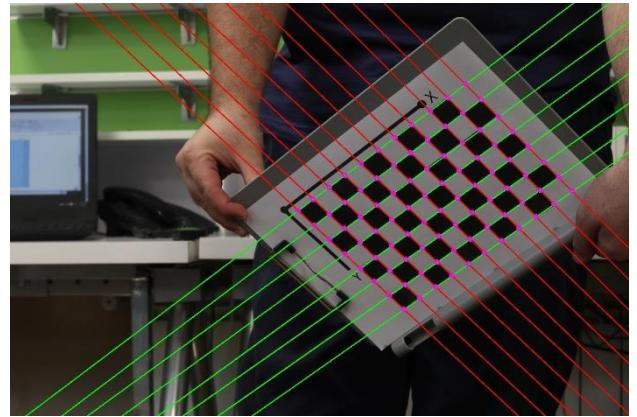
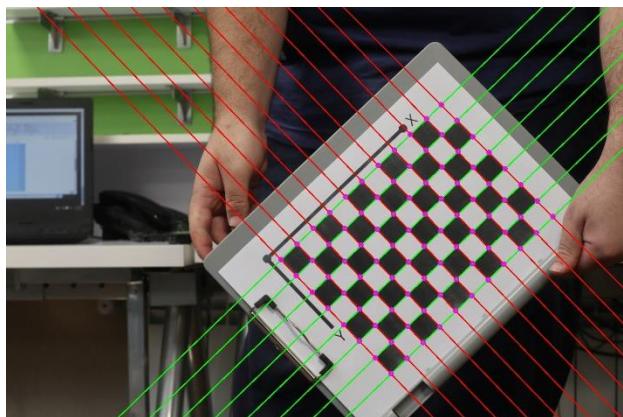
در صورتی که این نقطه برخورد، یک نقطه خوب باشد ، دو خطی که این نقطه برخورد را ایجاد میکردند را رسم میکنیم. و در غیر این صورت، هیچ یک را رسم نمیکنیم.

توجه داریم که تمام خطوطی که از ناحیه شطرنجی رد میشوند، حداقل یکبار رسم میشوند، زیر حداقل یکی از برخورد های آنها با سایر خطوط ناحیه شطرنجی، در تابع بالا قبول میشود.

اما سایر خطوط، هیچگاه رسم نمیشوند. زیر تک تک نقاط برخورد آنها با سایر خطوط دیگر در شرایط شطرنجی صدق نمیکند و در نتیجه هیچگاه رسم نمیشوند.



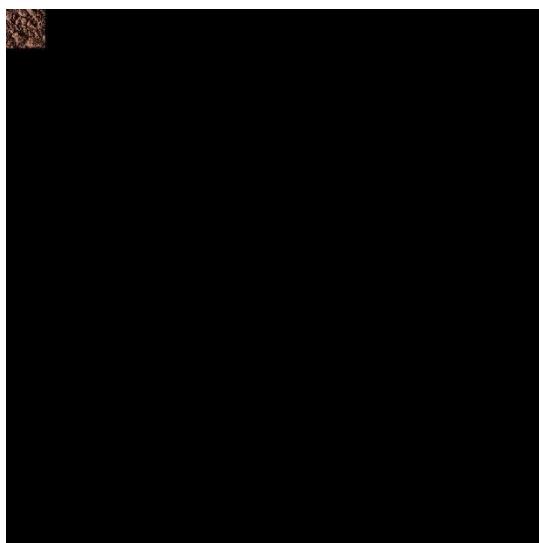
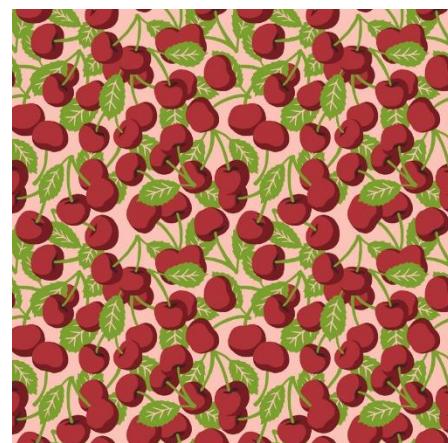
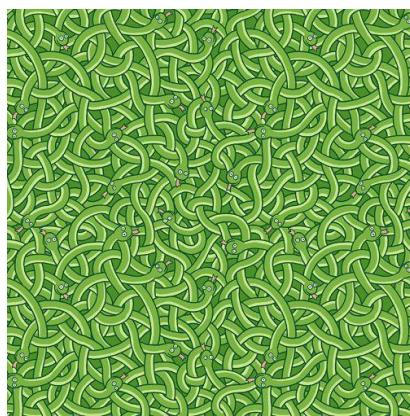
در نهایت، خطوطی که در مرحله بالا بدست آمده اند را مجدداً برخورد میدهیم (با فرمولی که ذکر شد) و با بدست اوردن نقاط برخورد، آنها را نیز رسم میکنیم.



توجه کنید که در قسمتی که میخواهیم خطوط مناسب ناحیه شترنجی را پیدا کنیم، این کار با توجه به شرایط دو عکس فعلی انجام شده و با تغییر عکس (افزایش روشنایی، تغییر کنتراست، تغییر بیش از حد زاویه کلاسور و ...) ممکن است نتیجه مناسبی بوجود نیاید.

سوال (۲)

تکسچر های انتخاب شده :



الگوریتم :

به طور پیش فرض، سایز هر بلوک را 185×185 پیکسل در نظر گرفته ایم. همچنین `overlap_size = 40` پیکسل قرار داده شده است. (شما میتوانید این اعداد را تغییر دهید اما ترجیح بر این است که از همین سایز های پیشفرض استفاده شود).

ابتدا یک بلوک به صورت رندم، از تکسچر انتخاب میشود و در بالا سمت چپ عکس نهایی، قرار میگیرد.



سپس از ردیف اول همین عکس، یک مربع 185×185 جدا میکنیم به صورتی که به اندازه overlap_size از انتهای بلوک اول رو در بر بگیرد.

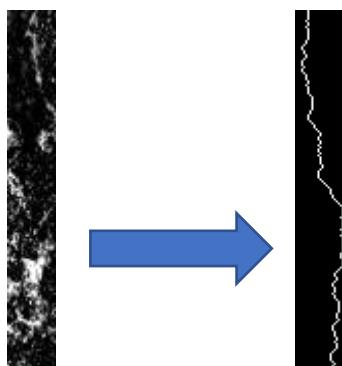


آنگاه یک ماسک سیاه و سفید به همین سایز به شکل روبرو میسازیم:

با استفاده از این ماسک و قسمت کراپ شده ، تابع best_match را صدا میزنیم. این تابع با استفاده از متند NCC و به کمک ماسک داده شده، بهترین مج ممکن با بیشترین مقدار شباهت را پیدا میکند و مختصات آن را

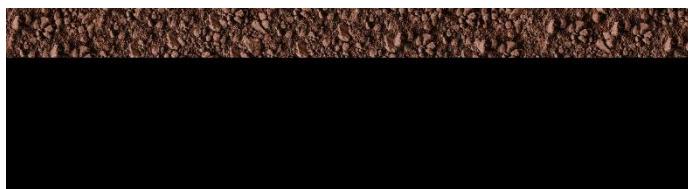
برمیگرداند. اینکار را به تعداد ۱۵ بار انجام میدهیم و ۱۵

نتیجه‌ی بهتر را در یک لیست میریزیم و مختصات یکی از آنها را به صورت رندوم برمیگرداییم.



سپس با استفاده از تابع row_min_path یک مسیر خوب در آن ناحیه اورلپ پیدا میکنیم:

هر چیزی که در سمت چپ این مسیر باشد، را از بلوک قبل(بلوک سمت چپ که قبلا جایگذاری شده) انتخاب میکنیم و هرچیزی که در سمت راست این مسیر باشد را از بلوک جدید انتخاب میکنیم. پس از جایگذاری این قسمت overlap ، مابقی بلوک جدید را نیز در جای خود قرار میدهیم.



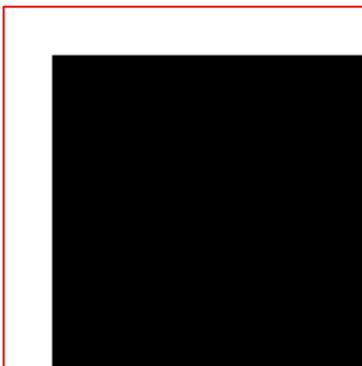
به همین ترتیب این حرکت را تا انتهای ردیف اول ، انجام میدهیم.



بعد از اتمام ردیف اول، حالا به سراغ ستون اول میرویم و همان رویه بالا رونجام میدهیم. با این تفاوت که این بار ناحیه های overlap ، به صورت مستطیل های افقی هستند و مسیر که در این ناحیه بدست می آید یک مسیر افقی است که هرچه در بالای مسیر باشد، از عکس قبلی(که تاکنون ساخته شده) انتخاب میشود و هرچه در زیر این مسیر باشد، از بلوک جدیدی که پیدا کرده بودیم بدست می آید. پس از اتمام ستون اول، شکلی مانند زیر حاصل شده است :

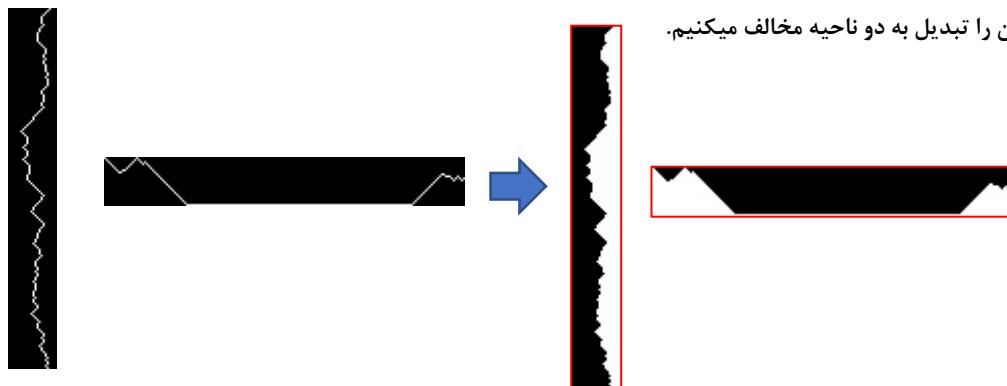


در نهایت، مابقی قسمت های عکس را تکمیل میکنیم. بلوک هایی که در این قسمت قرار میگیرند، با نواحی قبلی خود، یک ناحیه L شکل اشتراک دارند. (شکل سمت چپ)

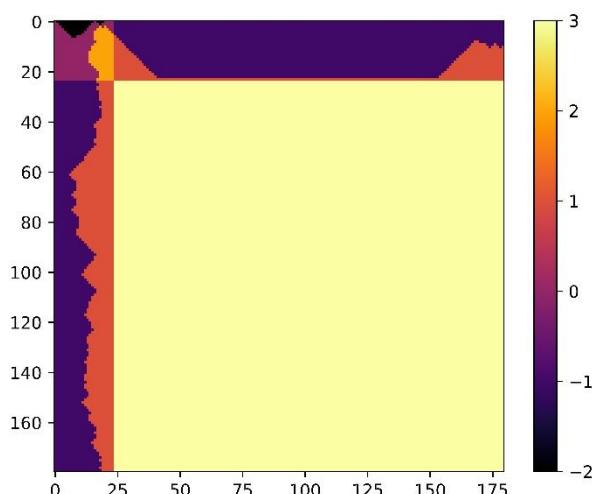


یک ماسک L شکل نیز میسازیم(شکل سمت راست)

مطابق قسمت های قبل، با استفاده از تمپلیت L شکل و ماسک L شکل، بهترین مج ممکن را به صورت رندوم پیدا میکنیم. حال باید این پنج را با عکس قبلی هماهنگ کنیم. برای این کار، باید ناحیه L بالا و ناحیه چپ بلوک پیدا شده را، با بقیه عکس مج کنیم. برای اینکار به طور مجزا بهترین مسیر افقی (برای بالای بلوک) و بهترین مسیر عمودی (برای سمت چپ بلوک) را پیدا میکنیم :



(تقریبا به صورت باینری، البته به این صورت هست که نواحی سفید رنگ برابر +1 و نواحی مشکی برابر -1 هستند)



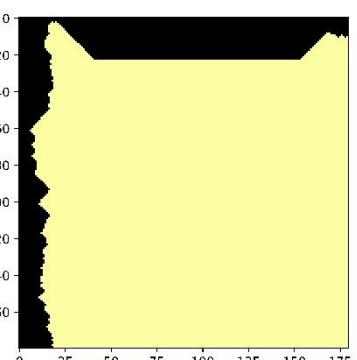
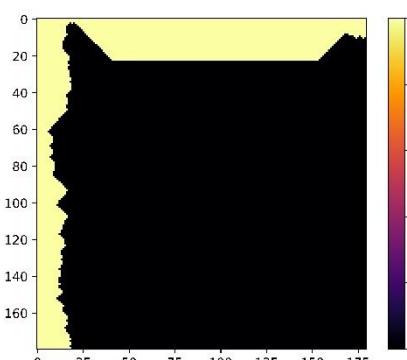
حال اگر یک ماتریس مربعی 185×185 جدید در نظر بگیریم (که در ابتدا تمام مقادیر آن 3 باشد) و این نوارهای پر شده را در بالا و چپ آن جایگذاری کنیم (اول نوار افقی رو میگذاریم و سپس نوار عمودی را با سمت چپ آن قسمت ماتریس، جمع میکنیم) یک ماتریس شبیه شکل روبرو بدست می آید).

که دارای مقادیر $+3, +2, +1, +0, -1, -2$ هست.

در این حالت اگر این شکل را به دو قسمت تبدیل کنیم:

۱. مجموعه شامل $+3, +2, +1$ را با هم.

۲. مجموعه $+0, -1, -2$ با هم



آنگاه شکلی مثل شکل راست بدست می آید (ماسک g₋₁)، شکل سمت چپ) این ماتریکس و معکوس آن (g₋₁، شکل سمت چپ) را به عنوان یک ماسک به کار خواهیم برد.

بنابراین کافی است ماسک سمت چپ را در قسمتی از عکس اصلی ضرب کنیم(یعنی قسمت هایی که میخواهیم از عکس اصلی نگه داریم) و ماسک سمت راست را در بهترین بلوک پیدا شده ضرب کنیم(قسمتی که میخواهیم از بهترین بلوک نگه داریم) و سپس این دو را با یک دیگر ترکیب میکنیم (جمع کنیم).



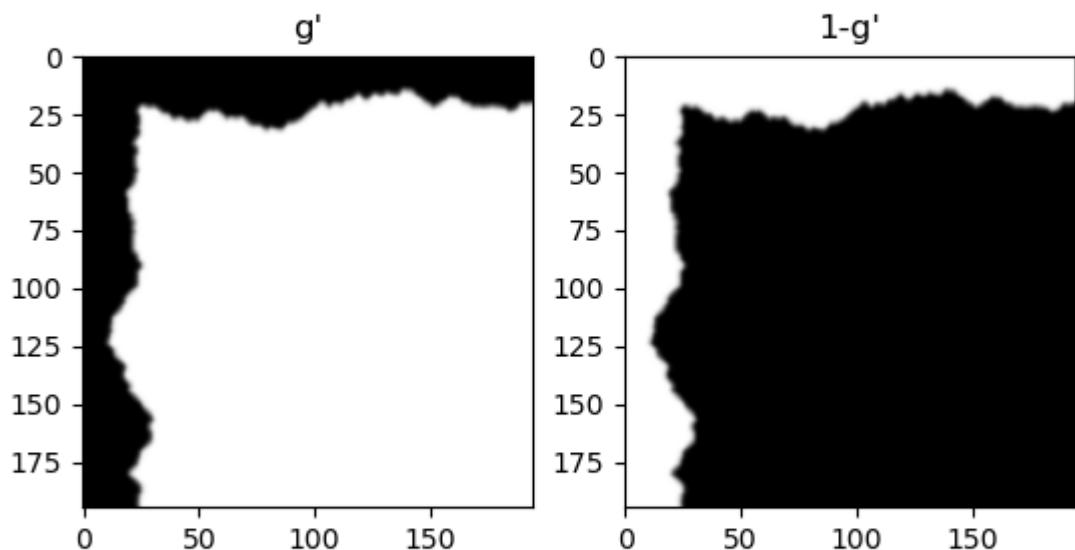
بهترین بلوک پیدا شده

تکه انتخاب شده از عکسی که
در مراحل قبل ساخته شده.

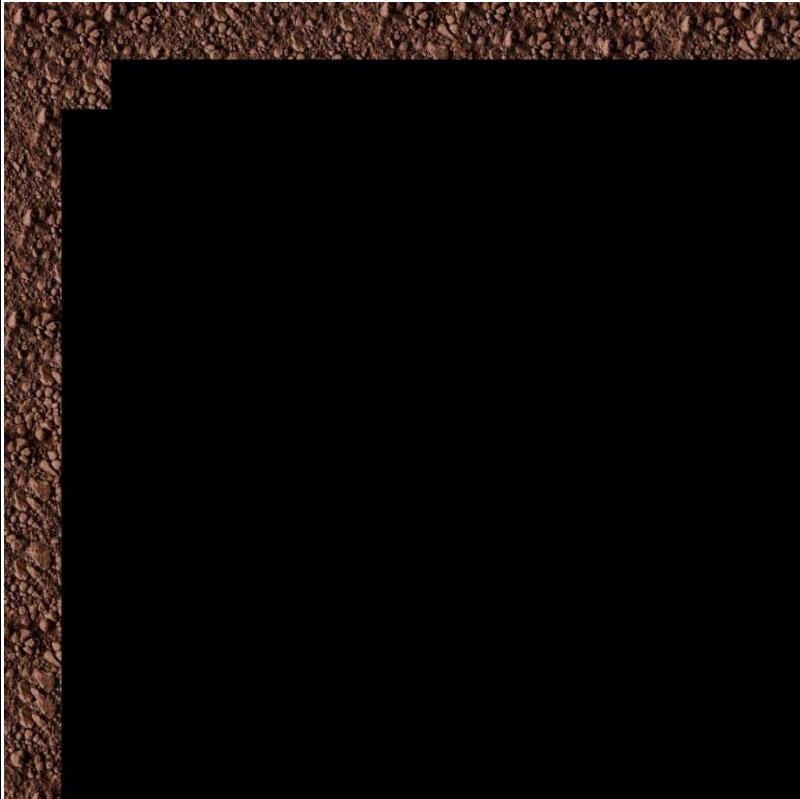
عکس حاصل از ترکیب دو
عکس دیگر

البته قبل از جمع زدن (چه این مرحله چه در مراحل پر کردن سطر یا ستون اول) ابتدا ماسک g بدست آمده را در یک فیلتر گاووسی 5×5 با سیگما ۱ ضرب میکنیم تا g' بدست بیاید. و سپس ماسک $1-g'$ را بدست میآوریم. با این کار، در قسمت مرز های سیاه و سفید ماسک، به صورت ناگهانی و هواهی انتخاب نمیکنیم، بلکه یک حالت ضربی وزنی بدست می اید که در ترکیب کردن دو عکس در این نواحی (نواحی نزدیک به بهترین مسیر) به صورت وزنی، بهتر عمل میکند.

Masks



در نهایت بلوک حاصل از جمع زدن وزنی را، در جای مناسب خود قرار میدهیم. برای مثال، پس از جایگذاری اولین بلوکی که در این حالت صدق میکند، به نتیجه زیر میرسیم.



و همین کار را برای سایر بلوک های باقی مانده نیز انجام میدهیم تا عکس تکمیل شود :



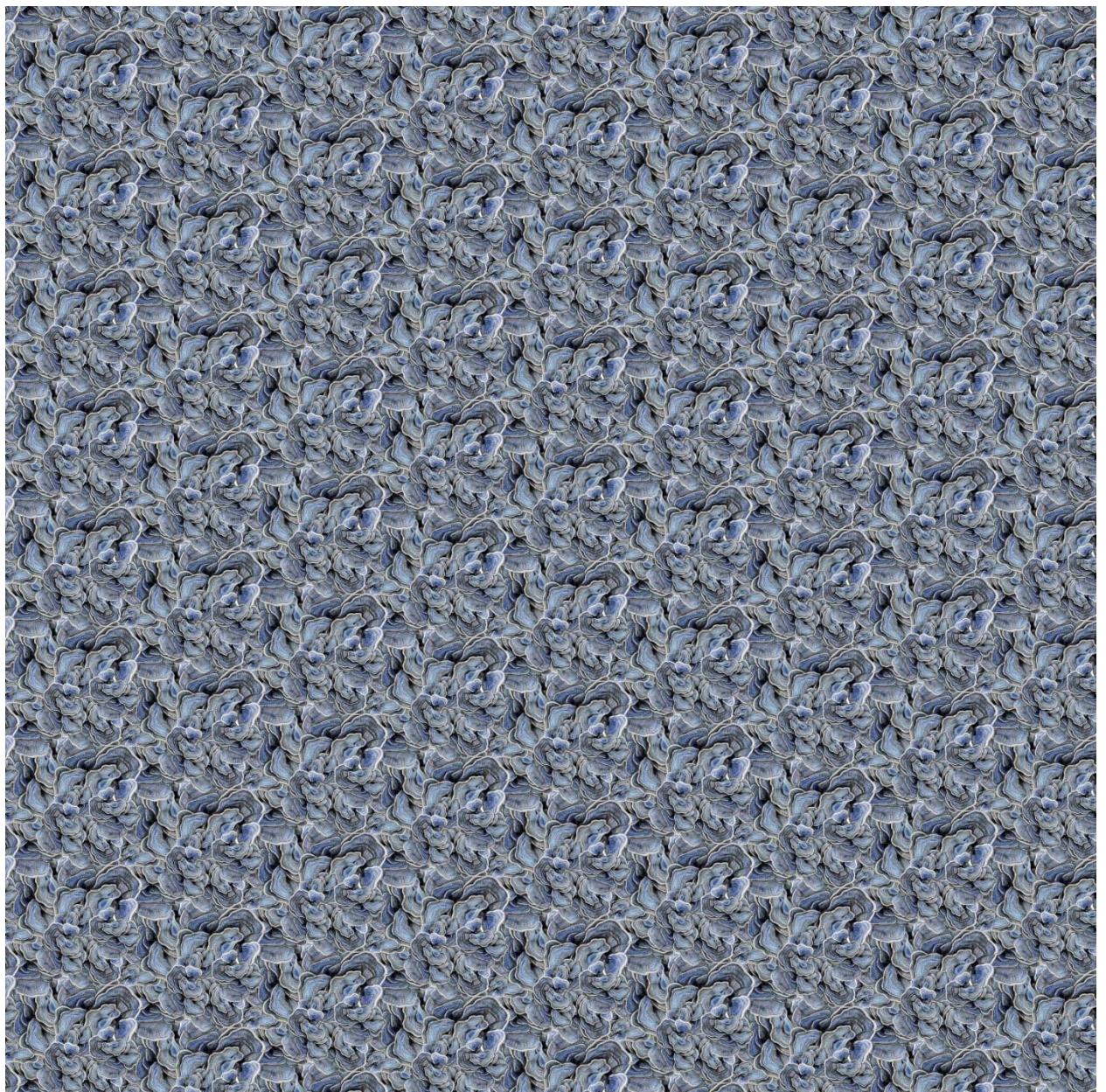
پ.ن : عکس های بالا ، طی اجرا های مختلف برنامه بدست آمده است و ممکن است دقیقا همه مربوط به یک اجرا باشند. اما کلیات عملکرد برنامه به همین صورت است.

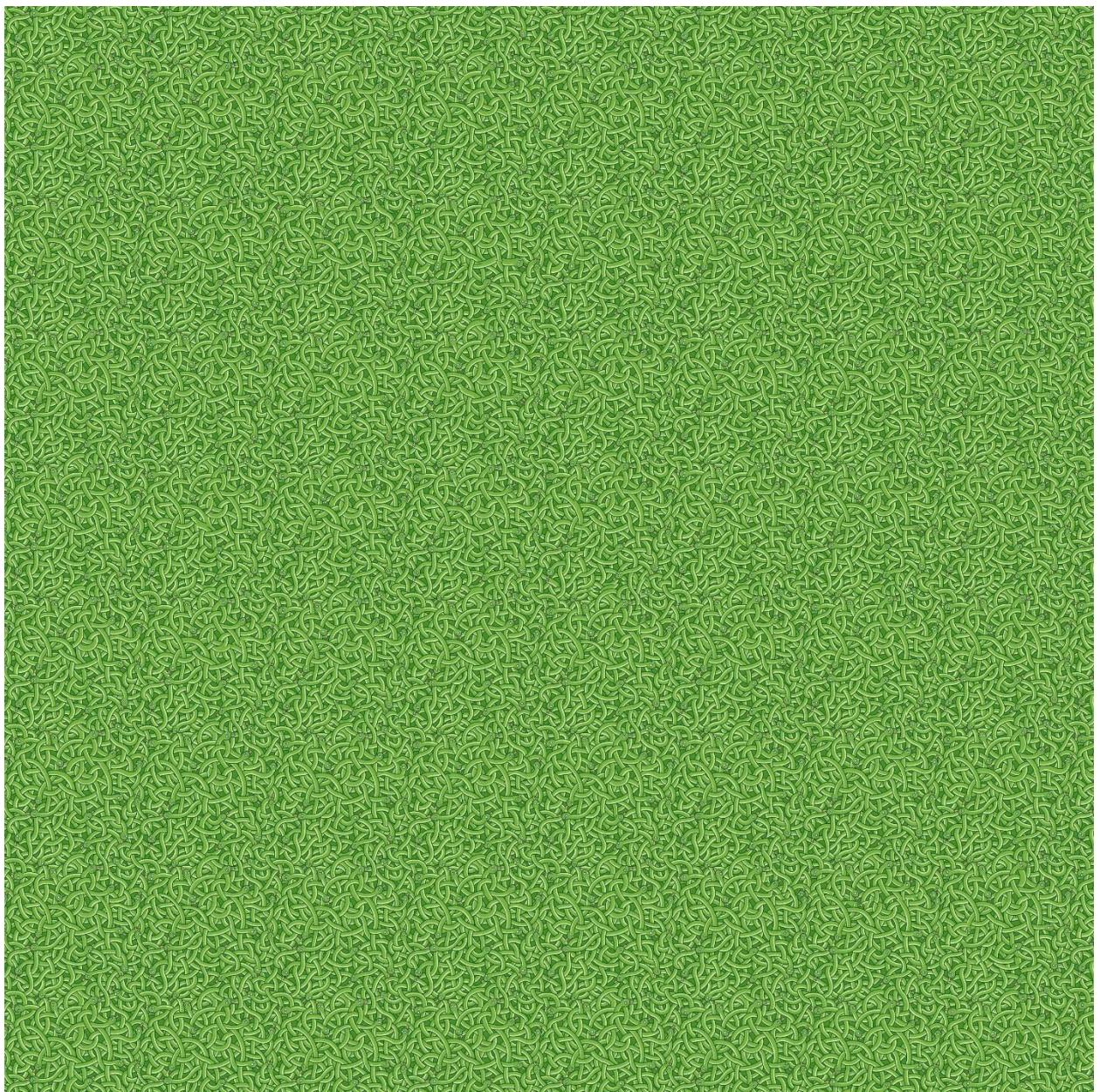
- برنامه این قابلیت را نیز دارد که از روند اجرای الگوریتم، فایل gif تهیه کند. برای استفاده از این قابلیت، بولین gif_enabled را فعال کنید.
- با تغییر مقدار percentage میتوانید سایز عکس های موجود در فایل گیف را تغییر دهید. به صورت پیشفرض، تمام عکس ها ۹۰٪ کوچک میشوند و سپس در فایل گیف ذخیره میشوند.
- برنامه این قابلیت را نیز دارد که مرحله به مرحله، نتایج را در عکس های مختلف ذخیره کند . برای استفاده از این قابلیت، بولین step_by_step را فعال کنید.
- در صورت استفاده از هر یک از قابلیت های بالا، نتایج متفرقه‌ی حاصل، در پوشه Result/q2 و در یک زیر‌فولدر مربوط به همان عکس قرار خواهد گرفت. **توجه شود که حجم نهایی این نتایج متفرقه، چیزی بین ۲ تا ۷ گیگابایت خواهد بود.**

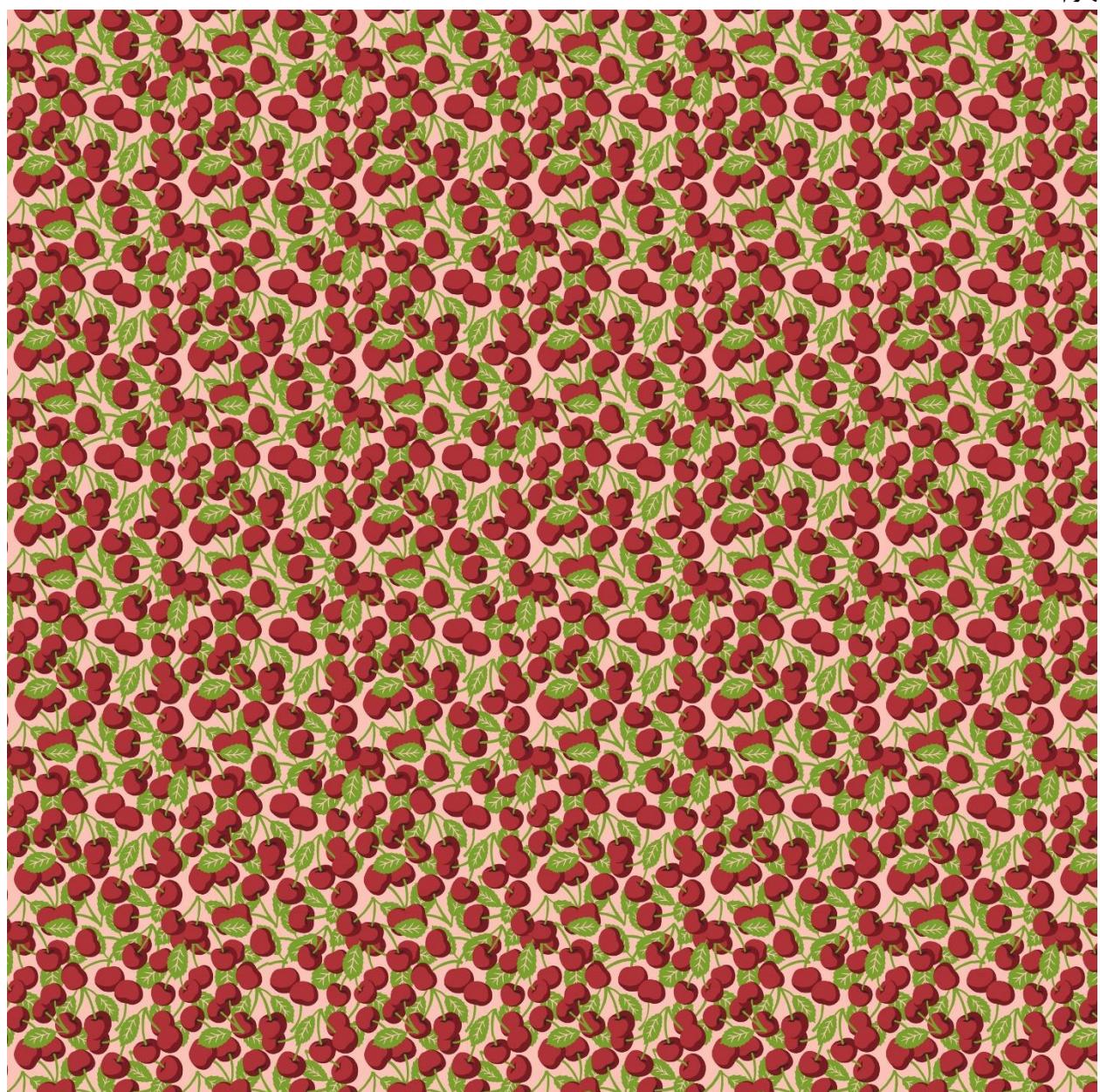
نتایج :

عکس اول :





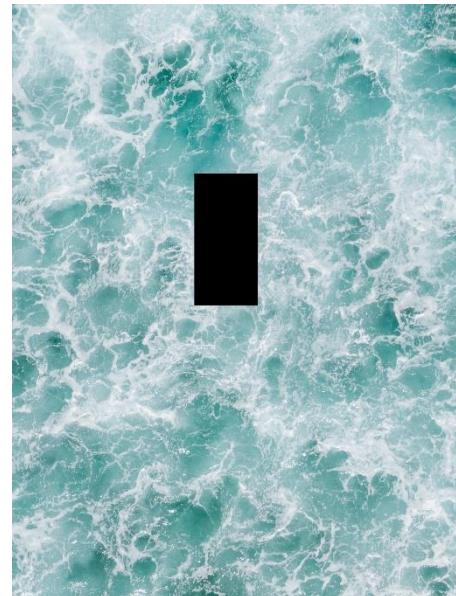




سوال سوم)

ورودی :

با استفاده از فتوشاپ، ناحیه های مورد نظر را سیاه میکنیم و مختصات آنها را نیز بدست می آوریم تا به صورت هاردکد در کدها استفاده شود :



روش کار، همان تکسچر سینتزايز تمرین دوم است. همانطور که در توضیح تمرین دوم گفته شد، در آن تمرین ابتدا سطر اول، سپس ستون اول و در انتهای بقیه عکس را پر میکنیم.

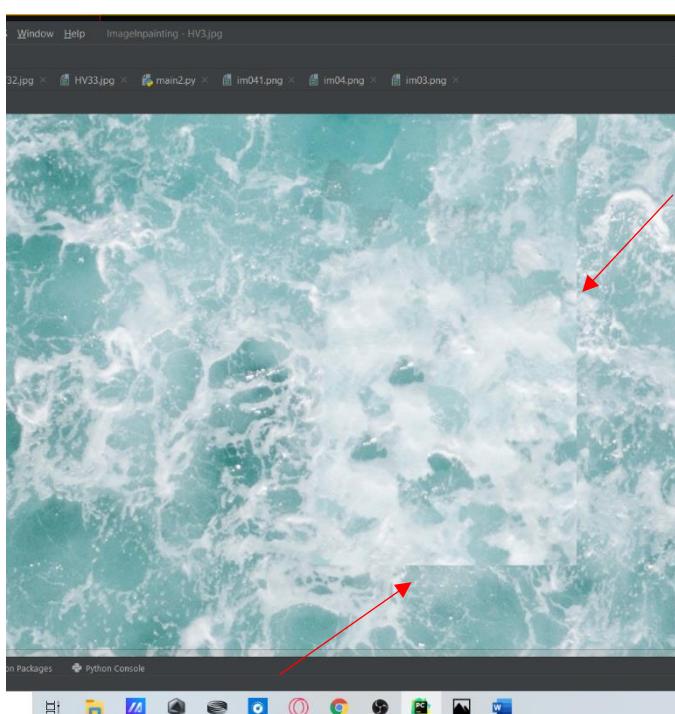
در این تمرین نیز مشابه قسمت سوم تمرین قبل عمل میکنیم (یعنی فرض میکنیم هر بلوکی که قصد پر کردن آن را داریم، با بقیه عکس یک اشتراک L شکل دارد).

روش کلی پر کردن در تمرین دوم توضیح داده شد و دیگر به آن نمیپردازیم. البته تفاوت کوچکی با تمرین قبلی داریم :

در این تمرین، برای پیدا کردن یک پچ خوب رندوم، شکل را به چندین قسمت تقسیم میکنیم (که هیچکدام شامل محدوده های سیاه رنگ نباشند زیر نمیخواهیم به اشتباه بلوکی از این قسمت ها انتخاب کنیم) سپس در هر پچ، چند پچ خوب را پیدا میکنیم. آنگاه از بین پچ های خوب پیدا شده، یکی را به صورت رندوم انتخاب و خروجی میدهیم.

پس از پر کردن ناحیه سیاه رنگ، چیزی شبیه شکل روبرو بوجود می آید :

چنانچه مشاهده میشود، این قسمت سیاه رنگ پوشده، به طور مناسبی از سمت چپ و بالا با سایر قسمت های عکس مج شده است. اما از طرف راست و پایین، اصلا با عکس هماهنگ نیست. (دلیل این موضوع این است که در تمرین دوم ما عمل سینتزايز کردن را از چپ به راست و از بالا به پایین انجام میدادیم و هیچگاه نیاز نبود که سمت راست و یا پایین یک بلوک را به چیزی مج کنیم).



برای رفع این مشکل نیز، کاری شبیه به مراحل قبل میکنیم. نیاز است که پایین و سمت راست این قسمت (که به راحتی میتوان یک خط تشخیص داد که دو طرف آن مشابه نیست) با بقیه عکس اصلی، مج شود.

عملیات مج کردن سمت راست تصویر:

۱. از نقطه شروع خط عمودی سمت راست، بلوک های 60×100 میبریم. این بلوک طوری انتخاب میشود که ۲۰ سطر ابتدایی از ۱۰۰ سطر آن، از بالا با بقیه ای عکس اشتراک داشته باشد، ۳۰ ستون سمت چپ آن با قسمت سمت چپ خط (یعنی قسمتی که در مرحله اول سینتزايز کردن بدست آمده و درون ناحیه سیاه رنگ بوده است). اشتراک داشته باشد و ۳۰ ستون راست آن، با سمت راست خط (که عکس اصلی است) اشتراک دارد.(در واقع انگار بلوک را طوری انتخاب کردیم که خط عمودی، آن را به دو قسمت تقسیم کنید).

۲. برای بلوک گفته شده، یک بهترین مج رندوم انتخاب میکنیم.

۳. به ازای مج پیدا شده و بلوک انتخاب شده، یک بهترین مسیر انتخاب میکنیم، این بهترین مسیر به طور خیلی حدودی n شکل یا چیزی شبیه به نیم دایره خواهد داشت :

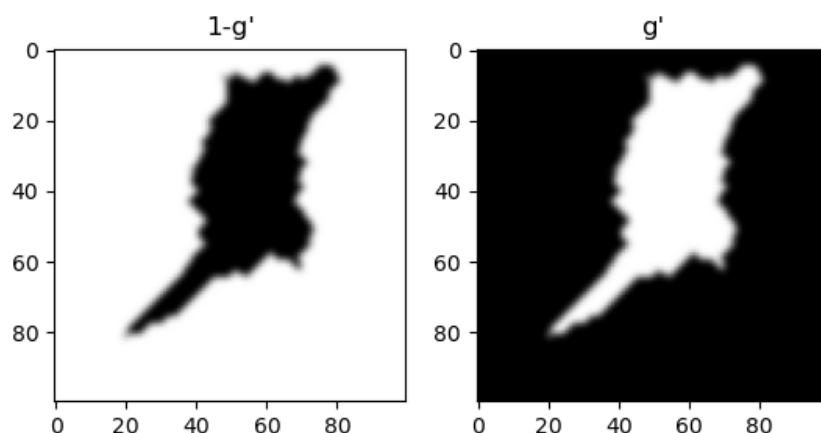
حال مطابق آنچه در تمرین دوم انجام داده بودیم، سپس یک فیلتر گاوی 5×5 با سیگما ۰ نیز به روی آن اعمال میکنیم (تا در ناحیه های مرزی، به طور تقریباً یکنواخت چگالی آن تغییر کند). و سپس معکوس این ماسک را نیز بدست می آوریم.

آنگاه ماسک سمت راست و قرینه آن را به ترتیب در بهترین مج رندوم انتخاب شده و بلوک انتخاب شده ضرب کرده، نتایج را با یکدیگر جمع میکنیم و آن را در ناحیه مورد نظر قرار میدهیم.

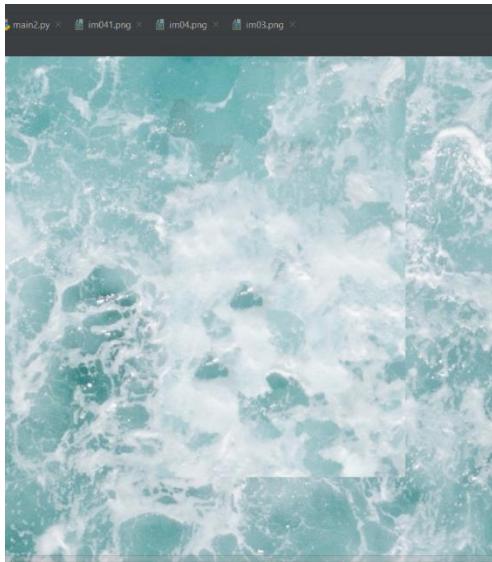
کاری مشابه همین کار را برای بایین عکس (خط افقی) انجام میدهیم. (البته در کد اصلی ابتدا خط افقی را درست میکنیم و سپس خط عمودی را، اینجا صرفا برای توضیح، خط عمودی را توضیح دادم).

در نهایت نیز، بلوک آخر(بایین راست ترین بلوک) را درست میکنیم. مرکز این بلوک محل برخورد دو خط عمودی و افقی است. این بلوک را باید از هر چهار جهت با بقیه عکس مج کنیم. روند مج کردن مشابه قبل است: یک بهترین مج رندوم پیدا میکنیم، ۲ ماسک درست میکنیم و یکی را در بهترین مج رندوم و دیگری را در آخرین بلوک ضرب میکنیم و این دو را ببا یکدیگر جمع میکنیم:

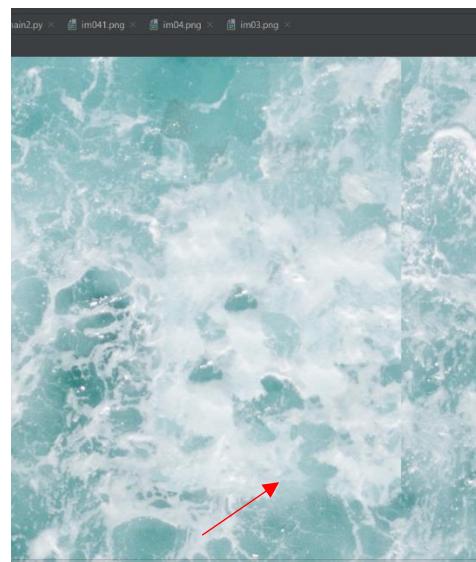
Masks



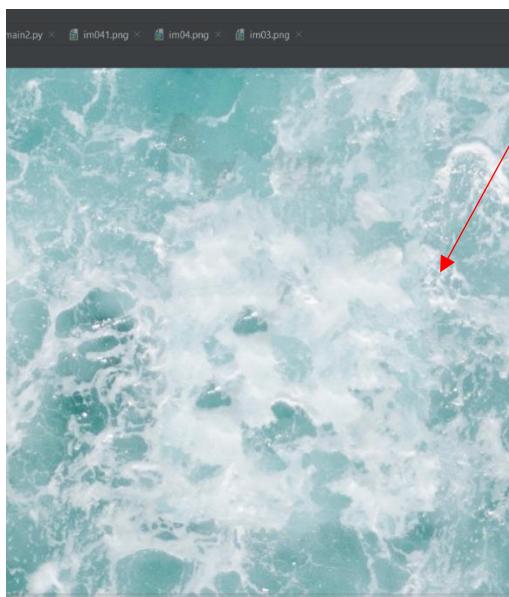
بعد از آن جام این روند :



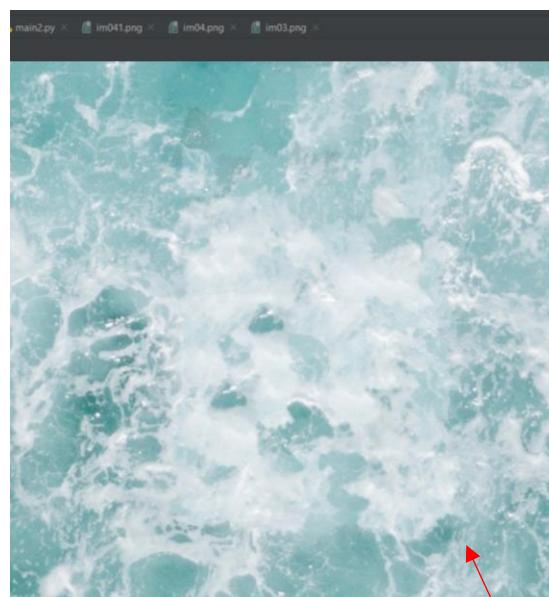
قبل از درست کردن سطر و ستون آخر



بعد از درست کردن سطر آخر

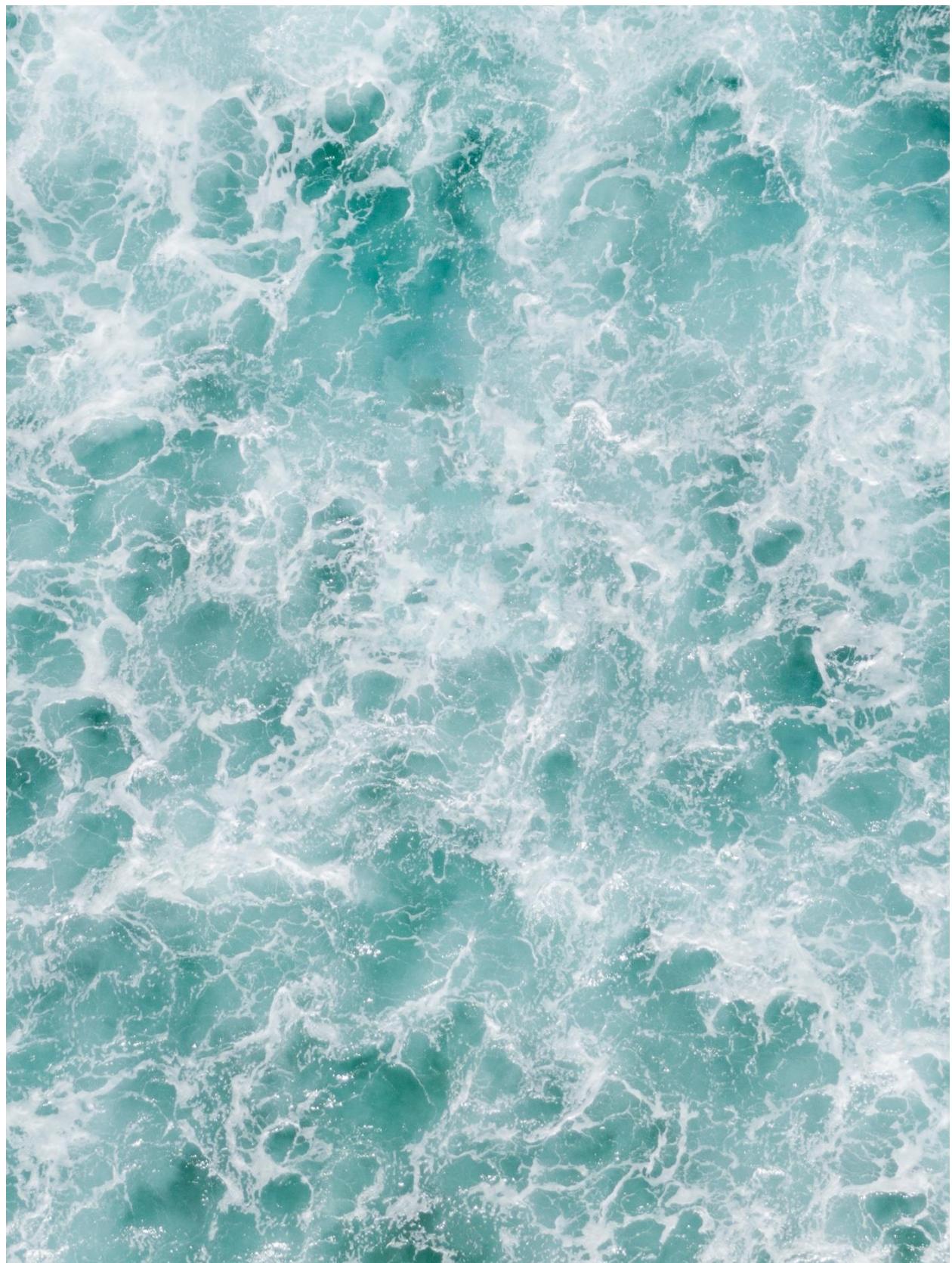


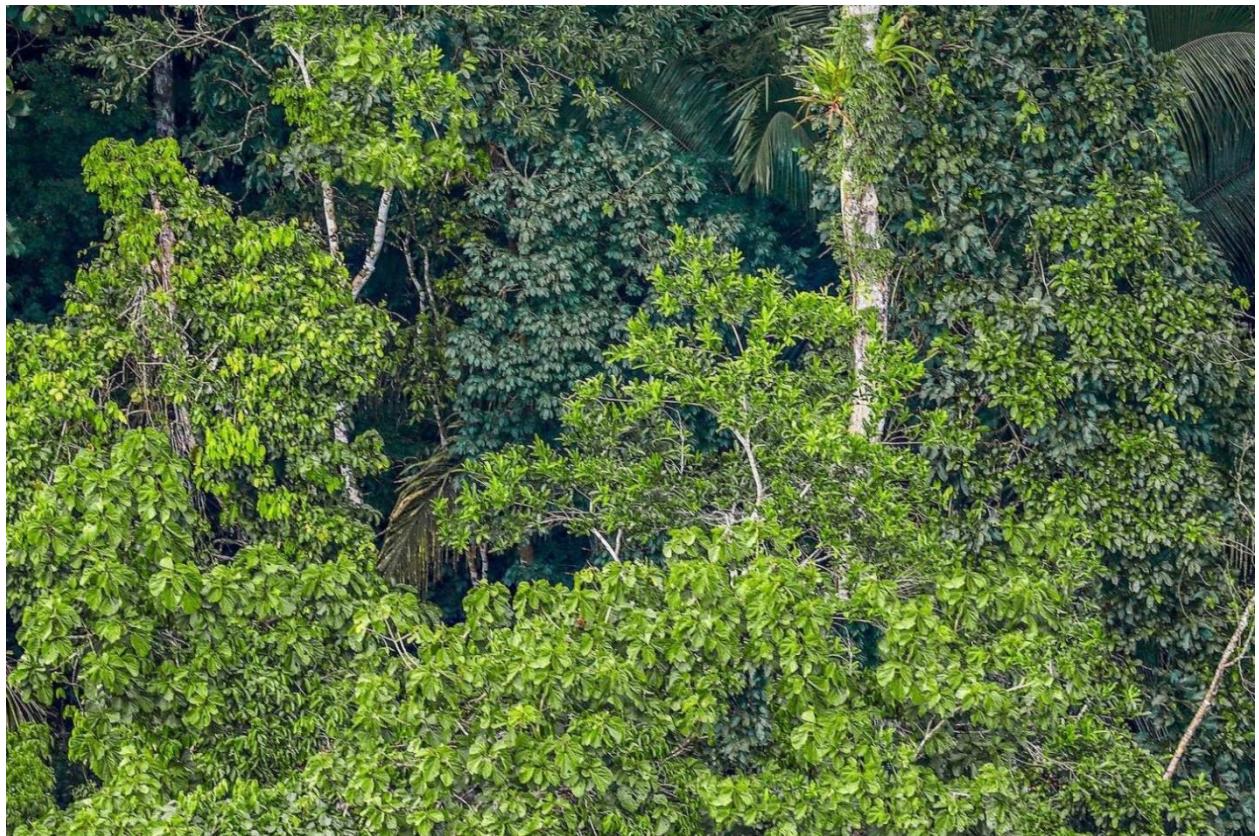
بعد از درست کردن سطر آخر و ستون آخر



بعد از درست کردن آخرین بلوک

عکس نهایی :





از آنجا که مقادیر عددی برای دو عکس اندگی متفاوت هستند ، کد هر کدام به طور جداگانه در کلاس های q3_birds.py و q3_sea.py پیاده سازی شده اند و شما با اجرای کلاس q3.py میتوانید آنها را اجرا کنید.