تمرین سری یک

درس تصویرپردازی رقمی

فرهاد دلیرانی

۹۶۱۳۱۱۲۵

dalirani@aut.ac.ir

dalirani.1373@gmail.com

فهرست

ابزارهای استفاده شده ۱
تمرین ۱ ..... ۲
قسمت a ۲
قسمت b ۲
قسمت C ۳
قسمت D ٤
قسمت E ٥
قسمت F ٦
قسمت G ۷
قسمت h ۹
قسمت i ۱۰
تمرین ۲ ..... ۱۲
قسمت a ۱۲
قسمت b ۱۲
قسمت c ۱۳
قسمت d ۱۳
قسمت e ۱٦
قسمت f ۱۸
قسمت g ۱۹
قسمت h ۲۰
قسمت i ۲۰
قسمت j ۲۱
قسمت k ۲٤
قسمت i ۲٦
تمرین ۳ ..... ۲۹
قسمت a ۲۹
قسمت b ۳۰
قسمت C ۳۰
قسمت d ۳۲
تمرین ٤ ..... ۳٤

قسمت a ٣٤
قسمت b ٣٤
قسمت c ٣٤
قسمت d ٣٥
قسمت e ٣٦
تمرین ٥ ..... ٣٧
قسمت a ٣٧
قسمت b ٣٨
تمرین ٦ ..... ٤٥
تمرین ٧ ..... ٤٧
قسمت a ٤٧
قسمت b ٤٩
قسمت c ٥١
تمرین ٨ ..... ٥٥
قسمت a ٥٥
قسمت b ٥٦
قسمت C ٥٧
قسمت D ٥٧

ابزارهای استفاده شده

زبان برنامه نویسی: زبان برنامه نویسی متلب

محیط توسعه: Matlab R2016a

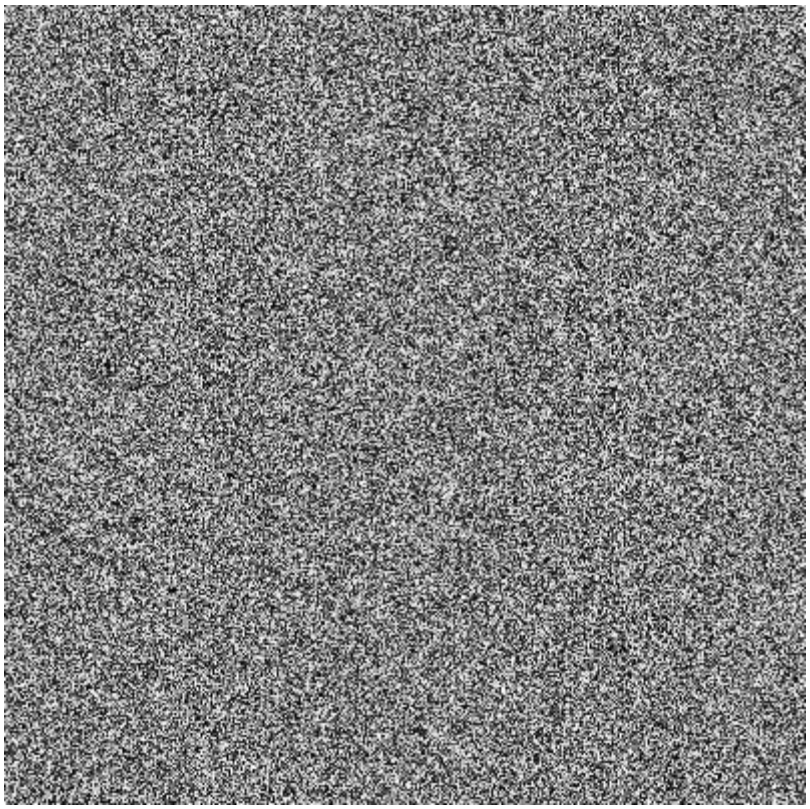
سیستم عامل: ویندوز ۱۰

تمرین ۱

کدهای این سوال در فولدر "p1" قرار دارند.

قسمت a

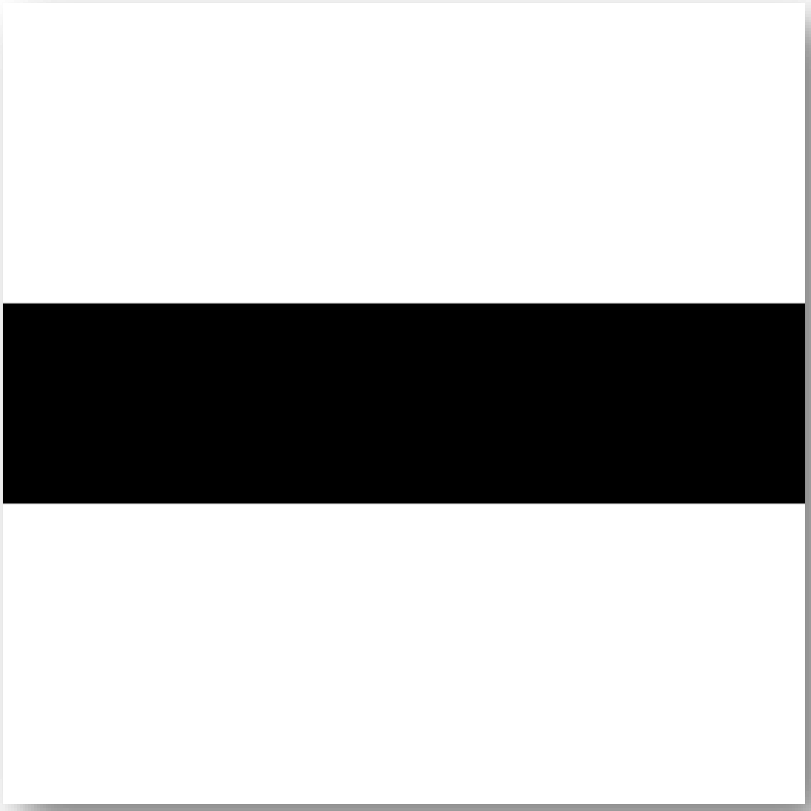
کد این قسمت در فایل p1a.m قرار دارد. برای این قسمت از سوال ابتدا با استفاده از تابع randi ، یک ماتریس ۴۰۰*۴۰۰ که هر درایه‌ی آن عددی صحیح از ۰ تا ۲۵۵ است ایجاد کرده‌ام سپس با استفاده از تابع uint8 تغییر دیتاتایپ مناسب را انجام داده‌ام و در آخر با استفاده از imshow و imwrite تصویر را نمایش داده‌ام و تصویر ایجاد شده را تحت نام p1a.png ذخیره کرده‌ام. در زیر تصویر خروجی را مشاهده می‌کنید:



قسمت b

کد این قسمت در فایل p1b.m قرار دارد. ابتدا یک ماتریس به ابعاد ۴۰۰*۴۰۰ ایجاد کرده‌ام و تمام درایه‌های آن را برابر ۲۵۵ (سفید) قرار داده‌ام. سپس مقدار تمام درایه‌هایی که در سطرهای ۱۵۱ تا ۲۵۰ هستند را برابر ۰

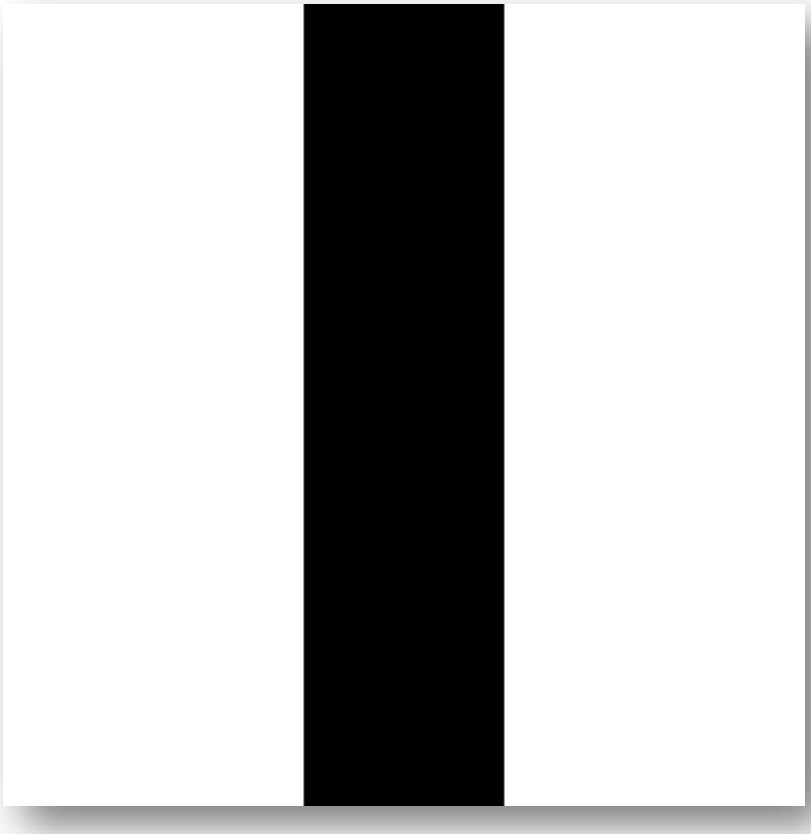
(سیاه) قرار داده‌ام. در انتها تصویر را نمایش داده‌ام و آن را تحت نام p1b.png ذخیره کرده‌ام. خروجی کد بالا نوشته شده برای این قسمت را در زیر مشاهده می‌کنید:



از آنجایی که در تصویر بالا پس زمینه سفید است و مرز تصویر مشخص نیست به تصویر بالا افکت سایه را اضافه کرده‌ام.

قسمت C

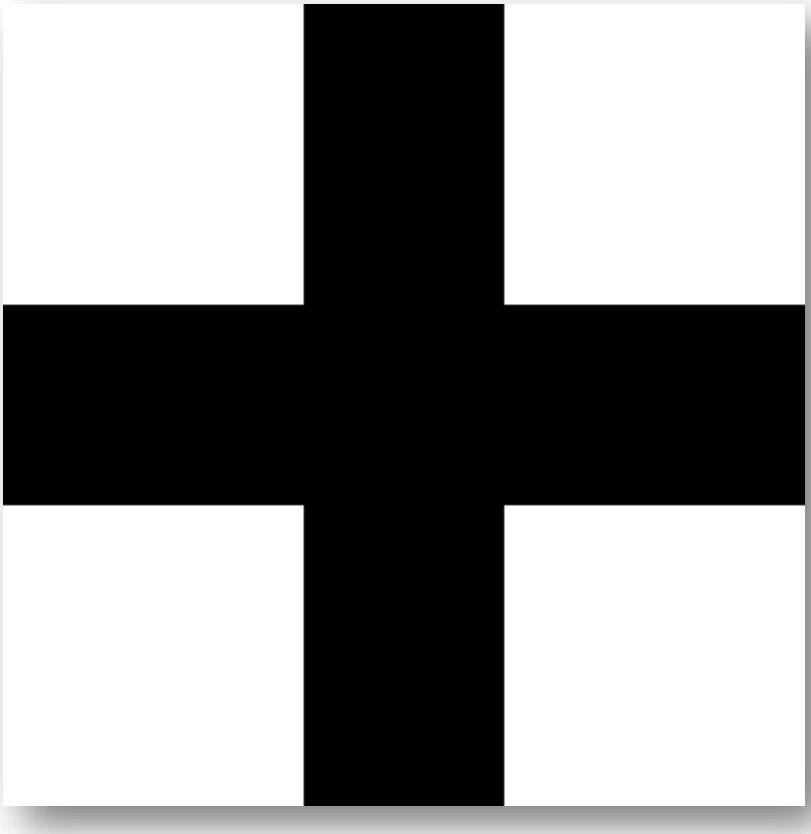
کد این قسمت در فایل p1c.m قرار دارد. ابتدا یک ماتریس به ابعاد ۴۰۰*۴۰۰ ایجاد کرده‌ام و تمام درایه‌های آن را برابر ۲۵۵ (سفید) قرار داده‌ام. سپس مقدار تمام درایه‌هایی که در ستون‌های ۱۵۱ تا ۲۵۰ هستند را برابر ۰ (سیاه) قرار داده‌ام. در انتها تصویر را نمایش داده‌ام و آن را تحت نام p1b.png ذخیره کرده‌ام. خروجی کد بالا نوشته شده برای این قسمت را در زیر مشاهده می‌کنید:



از آنجایی که در تصویر بالا پس زمینه سفید است و مرز تصویر مشخص نیست به تصویر بالا افکت سایه را اضافه کرده‌ام.

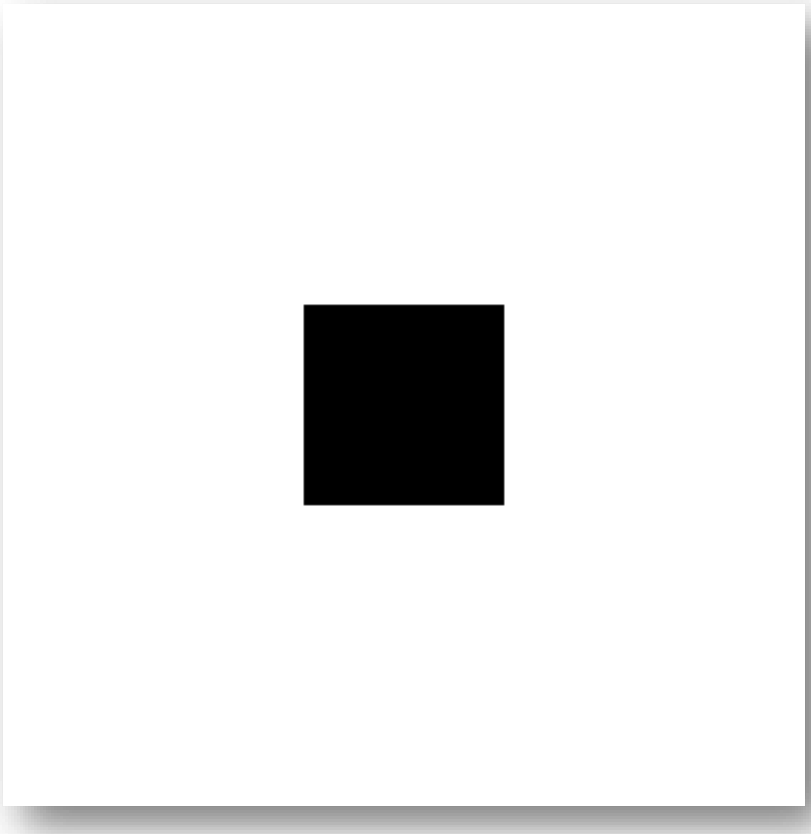
قسمت D

کدهای این قسمت از سوال در فایل p1d.m موجود است. در این سوال باید + تصویر بسازیم. برای ساخت علامت جمع ابتدا یک ماتریس ۴۰۰*۴۰۰ ایجاد کرده‌ام و تمام درایه‌های آن را برابر با ۲۵۵ قرار داده‌ام و بعد از آن درایه‌هایی که بین سطر ویا ستون‌های ۱۵۱ تا ۲۵۰ بوده‌اند را برابر با ۰ (سیاه) قرار داده‌ام. در انتها تصویر ایجاد شده را نمایش داده‌ام و آن را با نام p1d.png ذخیره کرده‌ام. در زیر تصویر + ایجاد شده را مشاهده می‌کنید:



قسمت E

کد این قسمت در فایل p1e.m موجود است. برای انجام این بخش ابتدا یک ماتریس ۴۰۰*۴۰۰ ایجاد کرده‌ام و در ابتدا مقدار تمام درایه‌های آن را برابر با ۲۵۵ (سفید) قرار داده‌ام. سپس درایه‌هایی که سطر و ستون‌شان بین ۱۵۱ تا ۲۵۰ است را به برابر ۰ (سیاه) قرار داده‌ام. سپس در انتهای کد آن را نمایش می‌دهم و آن را با نام p1e.png ذخیره کرده‌ام. در زیر تصویر ایجاد شده را مشاهده می‌کنید:



قسمت F

در این قسمت باید تصویر × نیز ایجاد کرد، کدهای این قسمت در فایل p1f.m موجود است. برای این کار ابتدا یک تصویر ۴۰۰*۴۰۰ ایجاد کرده‌ام و مقدار تمام درایه‌های آن را برابر با ۲۵۵ (سفید) قرار داده‌ام. معادله‌ی دو قطر تصویر که یک مربع ۴۰۰*۴۰۰ است، برابر با دو معادله‌ی زیر است:

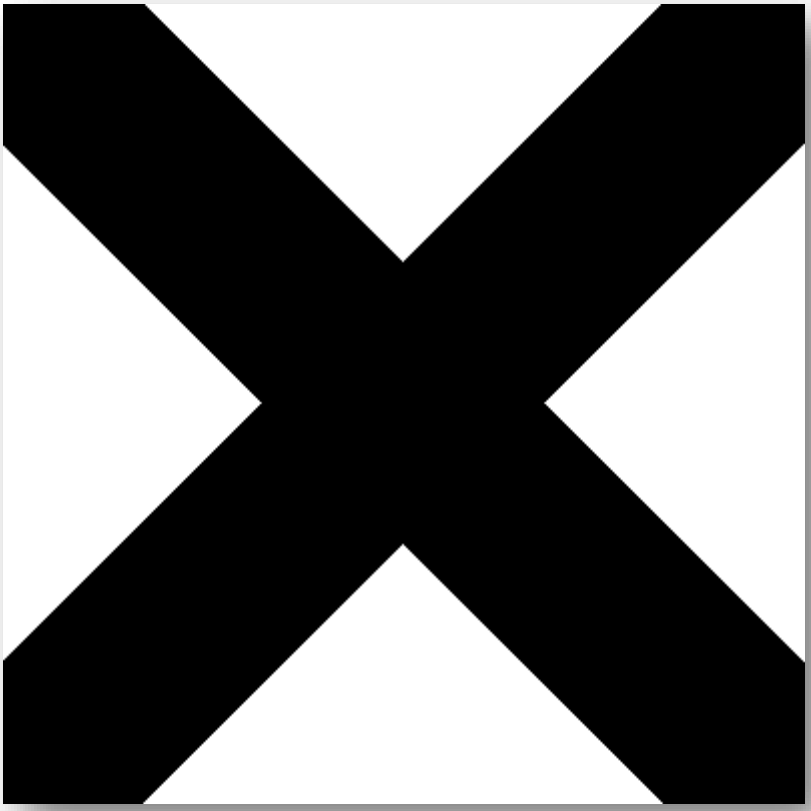
$$x + y - ۴۰۰ = ۰$$

$$x - y = ۰$$

همین طور فاصله‌ی نقطه $p(x.y.)$ تا خط $ax + by + c = ۰$ به صورت زیر به دست می‌آید:

$$\text{distance}(ax + by + c = 0, (x_0, y_0)) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

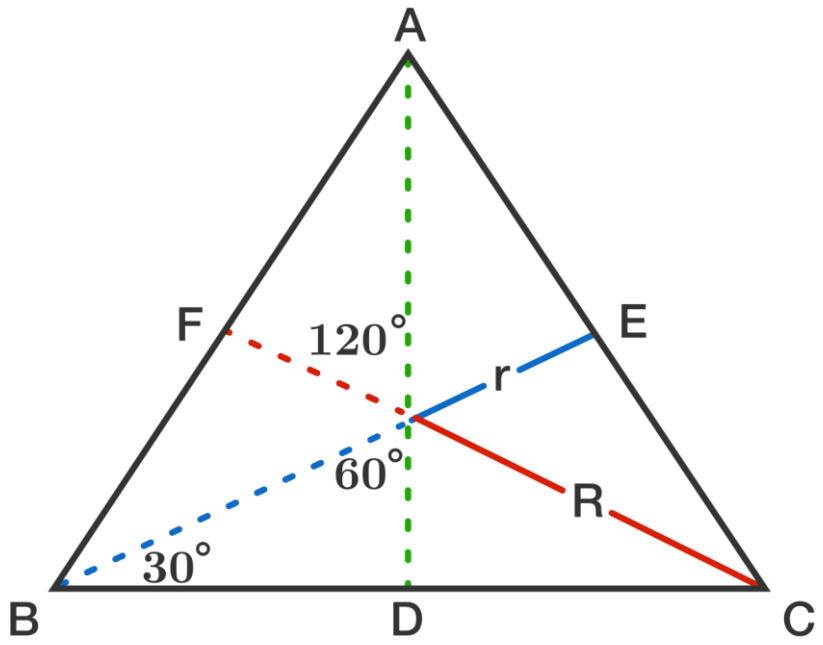
در نتیجه با استفاده از رابطه‌های بالا پیکسل‌هایی از تصویر که فاصله‌شان تا دو قطر کمتر از ۵۰ پیکسل است را برابر ۰ (سیاه) قرار می‌دهیم. در انتها تصویر ایجاد شده را نمایش داده‌ام و آن را با اسم p1d2.png ذخیره کرده‌ام. در زیر تصویر خروجی را مشاهده می‌کنید:



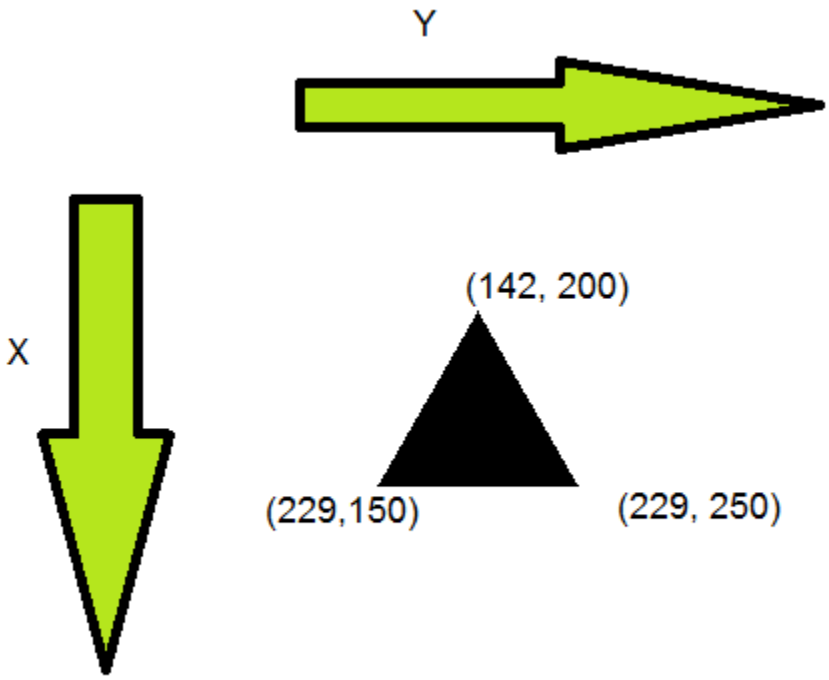
قسمت G

کد این قسمت از سوال در فایل p1g.m موجود است. هر ضلع مثلث متساوی الاضلاع خواسته شده برابر با ۱۰۰ است و باید در مرکز صفحه‌ی ۴۰۰ * ۴۰۰ قرار بگیرد.

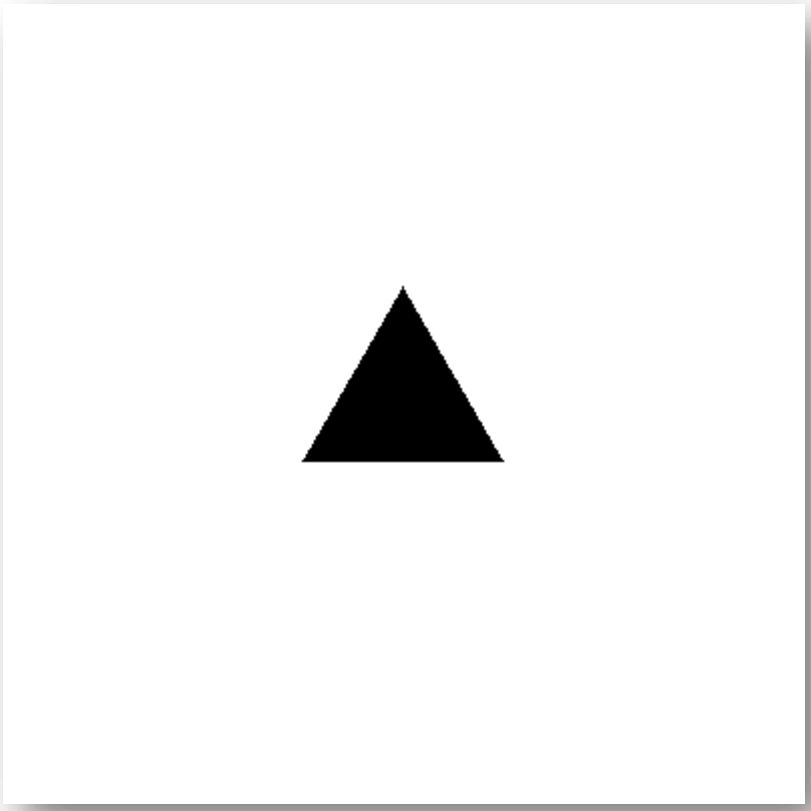
در زیر یک مثلث متساوی الاضلاع را مشاهده می‌کنید



با توجه مرکز مثلث که باید در نقطه‌ی (۲۰۰، ۲۰۰) مثلث قرار بگیرد و زاویه‌های درون مثلث مختصات نقطه‌های سه گوشه‌ی مثلث را به دست می‌آوریم که برابر با نقطه‌های زیر است:

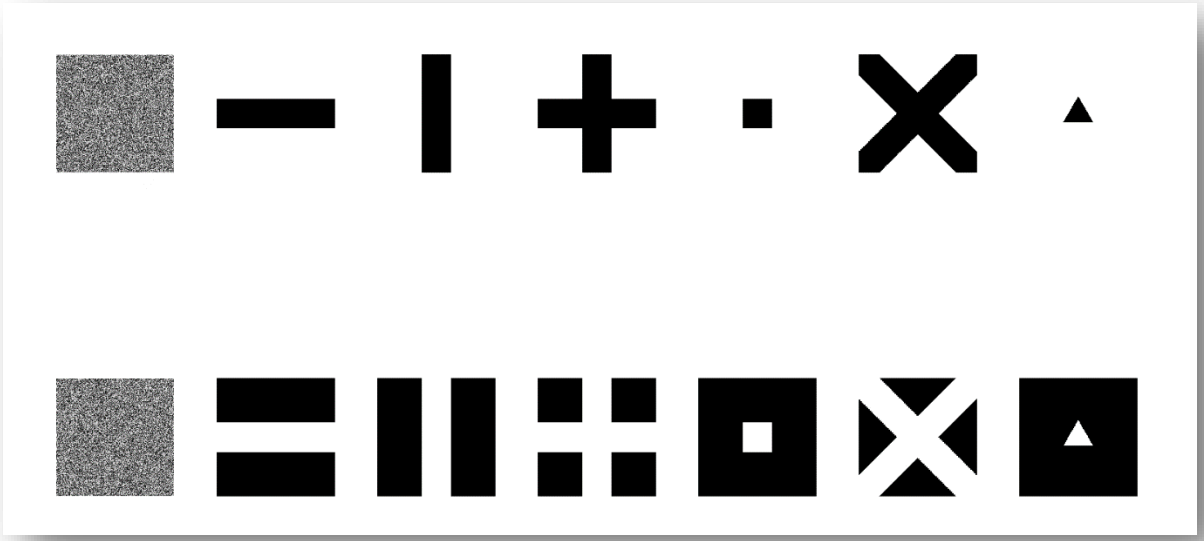


بعد از تشکیل یک ماتریس ۴۰۰*۴۰۰ نقطه‌هایی که در بین سه ضلع به دست آمده بین آن سه نقطه قرار می‌گیرند را به رنگ سیاه می‌آوریم. در آخر کد هم تصویر را نمایش داده‌ام و آن را با اسم p1g.m ذخیره کرده‌ام. در زیر تصویر خروجی را مشاهده می‌کنید:



قسمت h

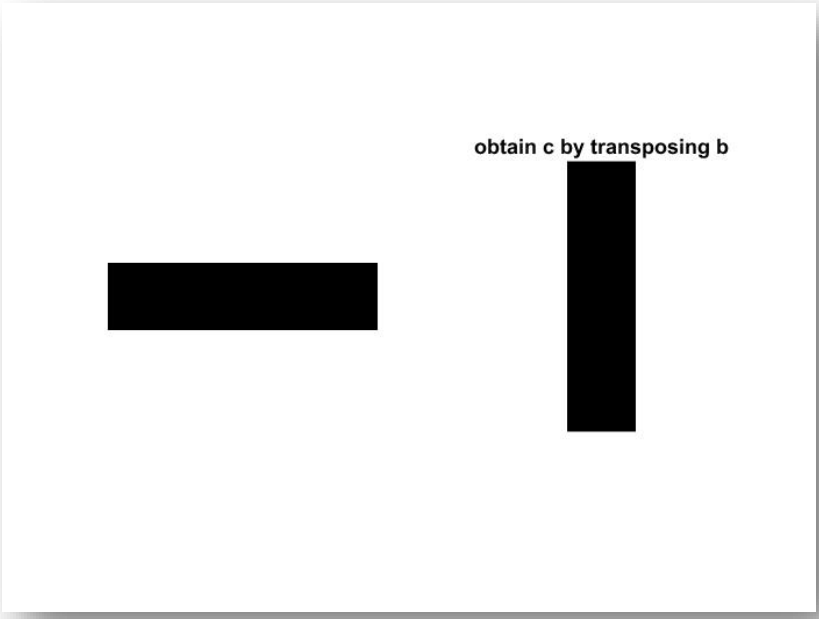
کدهای این قسمت در دوفایل p1h.m و p1h_run.m قرار دارند، برای تبدیل رنگ سیاه به سفید و برعکس، درایه‌های ماتریس را از ۲۵۵ کم کرده‌ام. تصویرها با پسوند png. موجود در پوشه‌ی کد را خوانده‌ام و به تابع داده‌ام. خروجی کدهای بالا را در تصویر زیر مشاهده می‌کنید:



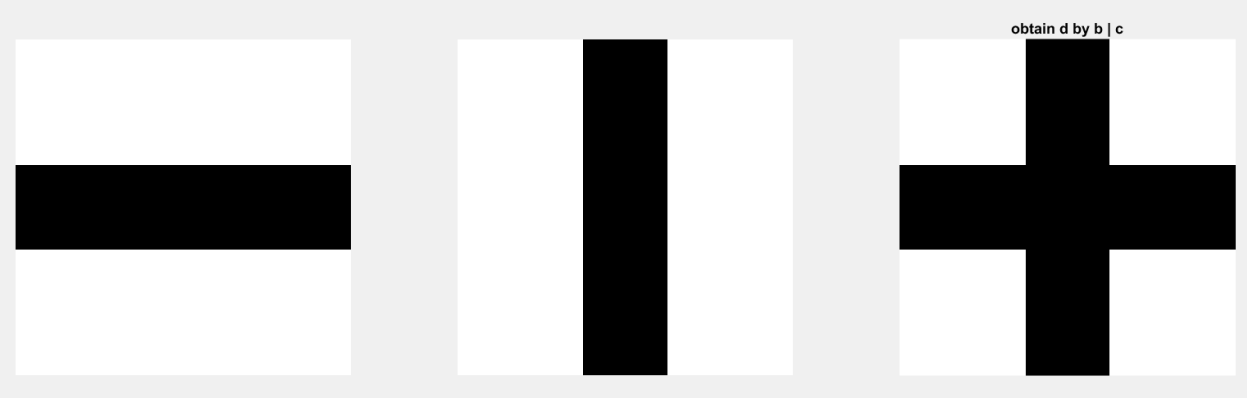
قسمت ز

کدهای این بخش در p1i.m قرار دارد. می‌توان بعضی از تصویرها را با انجام عملکردهایی از تصویرهای دیگر به دست آورد که در ادامه به آن‌ها می‌پردازم.

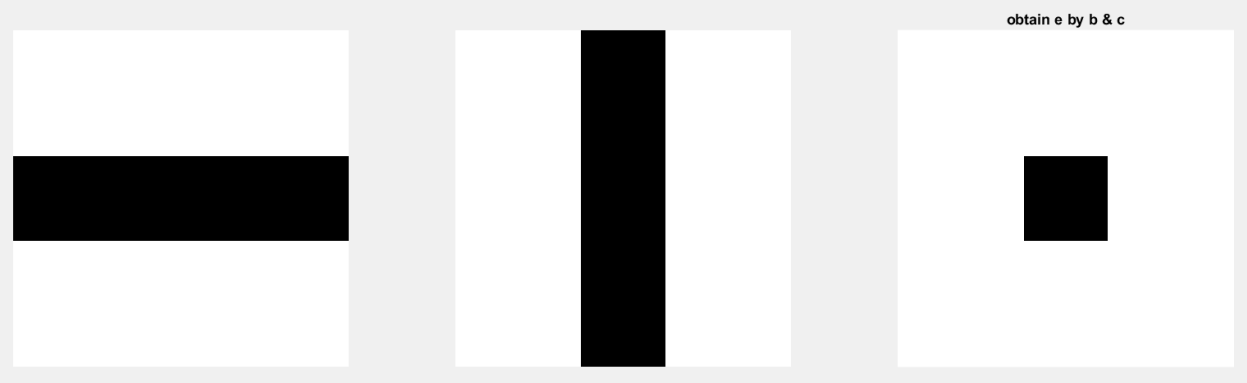
می‌توان با انجام ترانهاده بر روی ماتریس تصویر b به c رسید یا از با انجام ترانهاده بر روی ماتریس c به b رسید. در زیر تصویر خروجی حاصل از اعمال ترانهاده بر روی ماتریس b را مشاهده می‌کنید:



می‌توان تصویر d را که یک علامت به علاوه است را با and کردن دو تصویر b و c به دست آورد. در زیر خروجی حاصل از این کار را مشاهده می‌کنید.



می‌توان تصویر e را که یک علامت مربع است را با or کردن دو تصویر b و c به دست آورد. در زیر خروجی حاصل از این کار را مشاهده می‌کنید.



تمرین ۲

قسمت a

کد این قسمت در فایل p2a.m موجود است. در ابتدا با استفاده از دستور imread تصویر را خوانده‌ام و سپس با استفاده از imshow آن را نمایش داده‌ام. در زیر تصویر نمایش داده شده را مشاهده می‌کنید:

قسمت b

کد این قسمت در فایل p2b.m موجود است. با استفاده از دستور tgb2gray تصویر رنگی را به grayscale کرده‌ام. در زیر خروجی این کار را مشاهده می‌کنید:

قسمت c

کد این قسمت در p2c.m قرار دارد. در این قسمت با استفاده از دستور imcrop یک قطعه‌ی ۵۰*۵۰ از تصویر جدا کرده‌ام که گوشه‌ی سمت چپ مربع در نقطه‌ی (۱۷۰, ۳۷۰) قرار دارد. خروجی این کار را در تصویر زیر مشاهده می‌کنید:

قسمت d

کدهای این قسمت در فایل p2d.m قرار دارد. در این قسمت با استفاده از دستوری imrotate تصویر را ۹۰، ۱۸۰، ۲۷۰ درجه چرخش داده‌ام و بعد از آن ها را ذخیره کرده‌ام و همین‌طور نمایش داده‌ام. در زیر خروجی چرخش‌های مختلف را مشاهده می‌کنید:

۹۰ درجه چرخش:

۱۸۰ درجه چرخش:

۲۷۰ درجه چرخش:

قسمت e

کدهای این قسمت در p2e.m موجود است. در این قسمت از دستور imresize با فکتور ۰,۵ و ۲ استفاده کرده‌ام تا تصویرها را اسکیل کنم. در زیر خروجی‌های حاصل از کد را در زیر می‌بینید:

اسکیل با ضریب ۰,۵:

اسکیل با ضریب ۲:

قسمت f

کدهای این قسمت در p2f.m قرار دارد. در این قسمت از دستور flipdim با آرگمان ۲ برای ایجاد Horizontal Flip استفاده کرده‌ام و همین طور از دستور flipdim با آرگمان ۱ برای ایجاد Vertical Flip استفاده کرده‌ام.

در انتهای کد تصویرهای ایجاد شده را نمایش داده‌ام و آن‌ها را ذخیره کرده‌ام. در زیر خروجی‌ها را مشاهده می‌کنید:

Horizontal Flip:

Vertical Flip:

قسمت g

کد این قسمت در فایل p2g.m قرار دارد. در این قسمت دو ماتریس T2 و T3 را کنار هم قرار داده‌ام. در زیر خروجی این کار را مشاهده می‌کنید:

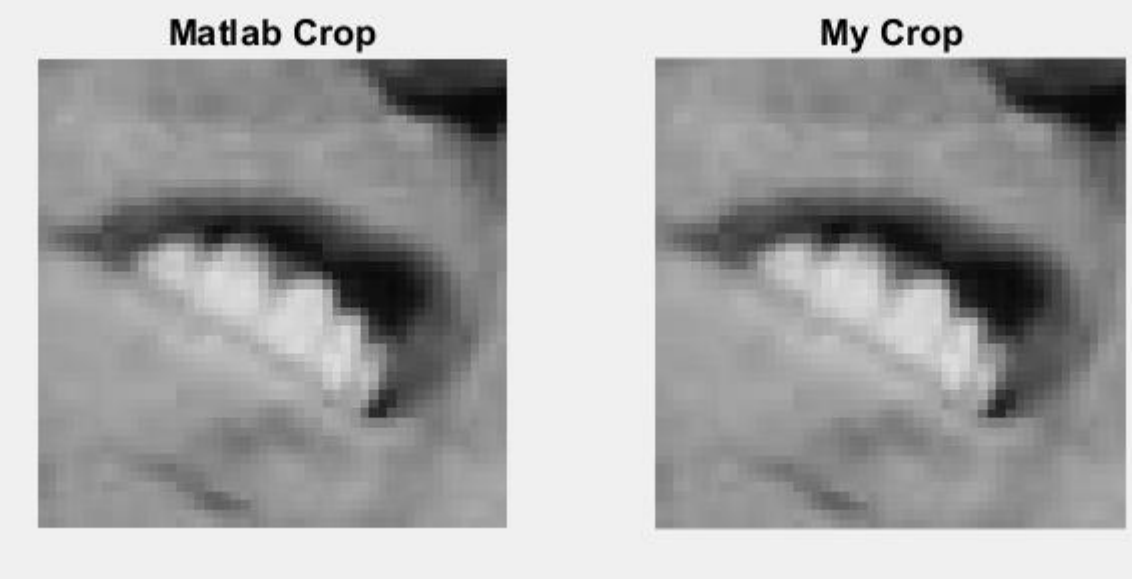
قسمت h

کدهای این قسمت در p2h_func.m و p2h_run.m قرار دارد. در تابع‌ای که در فایل p۲□□□□□□□.□ نوشته‌ام برای تبدیل rgb به gray scale سه طیف قرمز، سبز، آبی تصویر را جمع کرده‌ام و بعد آن را تقسیم بر سه کرده‌ام. در زیر خروجی تابع من و تابع متلب را مشاهده می‌کنید:

با اینکه از نظر دیدن با چشم دو تصویر یکسان هستند ولی با از تابع isequal متلب در آخرین خط کد متوجه شدم دو تصویر یکسان نیستند.

قسمت j

کدهای این قسمت در p2i_func.m و p2i_run.m قرار دارد. در این قسمت تابع برش تصویر خود را نوشته‌ام که با استفاده از آن، یک مربع ۵۰*۵۰ که گوشه‌ی بالا سمت چپ آن در مختصات (۳۷۰و۱۷۰) است را از تصویر می‌برم. خروجی این قسمت و مقایسه‌اش را با تابع متلب مشاهده می‌کنید:



در خط آخر کد از تابع isequal استفاده کرده‌ام که نشان می‌دهد دو تصویر یکسان هستند.

قسمت j

کد این قسمت در p2j_func.m و p2j_run.m قرار دارد. در تابع‌ای که برای چرخش تصویر با زاویه‌های ۹۰، ۱۸۰، ۲۷۰ نوشته‌ام اگر میزان چرخش ۹۰ درجه باشد ابتدا ترانهاده‌ی تصویر را به دست می‌آورم سپس تصویر را vertical flip می‌کنم. اگر میزان چرخش ۱۸۰ درجه باشد ابتدا تصویر را vertical flip می‌کنم سپس آن را horizontal flip می‌کنم. اگر میزان چرخش ۲۷۰ درجه باشد ابتدا ترانهاده تصویر را به دست می‌آورم سپس آن را horizontal flip می‌کنم. نتایج به دست آمده را در زیر مشاهده می‌کنید:

چرخش ۹۰ درجه:

چرخش ۱۸۰ درجه:

چرخش ۲۷۰ درجه:

همین طور که مشاهده می‌شود خروجی‌ها با خروجی‌های تابع چرخش متلب برابر است. در سه خط آخر کد از تابع isequal متلب استفاده کرده‌ام و سه تصویر چرخش داده شده توسط تابع خودم را با تابع متلب مقایسه کرده‌ام که خروجی isequalها نشان می‌دهد آن‌ها برابراند.

قسمت k

کدهای این قسمت در p2k_func و p2k_run.m قرار دارد. در p2k_func.m تابعی نوشته‌ام که تصاویر را با توجه به آرگمان ورودی دوبرابر یا نیم برابر می‌کند. برای نیم برابر کردن تصویر یک تصویر با طول و عرض نصف

تصویر اصلی ایجاد کرده‌ام و پیکسل‌های هر سطر را یک درمیان انتخاب کرده‌ام و بعد از آن سطرهای جدید ایجاد شده را یک درمیان انتخاب کرده‌ام و در تصویر جدید گذاشته‌ام.

برای دو برابر تصویر یک تصویر جدید با طول و عرض دو برابر تصویر جدید ایجاد کرده ام و به طور کلی این کار را انجام داده‌ام: برای این کار هر سطر ماتریس ورودی را گرفته ام و بین هر پیکسل و پیکس ل بعدیش در هر سطر یک پیکسل قرار داده ام و آن پیکسل را با میانگین دو پیکسل کنارش پر کرده‌ام. سپس سطرها را یک در میان در تصویر چیده‌ام به طوری که میان هر دو سطر جدید ساخته شده یک سطر خالی قرار دارد که میانگین دو سطر بالا و پایین آن سطر را در آن سطر قرار داده‌ام و اینگونه تصویر را دو برابر کرده‌ام و در ادامه‌ی کد خروجی‌ها را نمایش داده‌ام و آن‌ها را ذخیره کرده‌ام در ادامه خروجی این کار را مشاهده می‌کنید(بر اساس محدودیت عرض فایل ورد ممکن است تصویر دوم در اندازه‌ی واقعی نمایش داده نشده باشد. به تصویر خروجی موجود در فایل سوال مراجعه شود)

ضریب ۰,۵

ضریب ۲

همین طور که مشاهده می‌شود وقتی به صورت شهودی به تصویرها نگاه می‌کنیم تفاوتی به خروجی‌های من و تابع متلب نیست ولی در آخر کد از دستور isequal استفاده کرده‌ام که بر اساس آن خروجی تابع من و متلب یکسان نیست.

قسمت j

کدهای این قسمت در فایل‌های p2l_func.m و p2l_run.m قرار دارد. در فایل p2l_func.m تابع معادل flipdim متلب قرار دارد. برای vertical flip جای سطر i و n-i+1 را عوض کرده‌ام که n تعداد سطرها است. برای horizontal flip جای ستون i و m-i+1 را عوض کرده‌ام که m تعداد ستون‌ها است.

خروجی‌های کد را در زیر مشاهده می‌کنید:

Horizontally Flip:

vertical Flip:

در انتهای کد، isequal را استفاده کرده‌ام تا خروجی‌های خود را با تابع‌های متلب مقایسه کنم. طبق خروجی isequal تابع نوشته شده با تابع متلب عملکرد مشابه‌ای دارد.

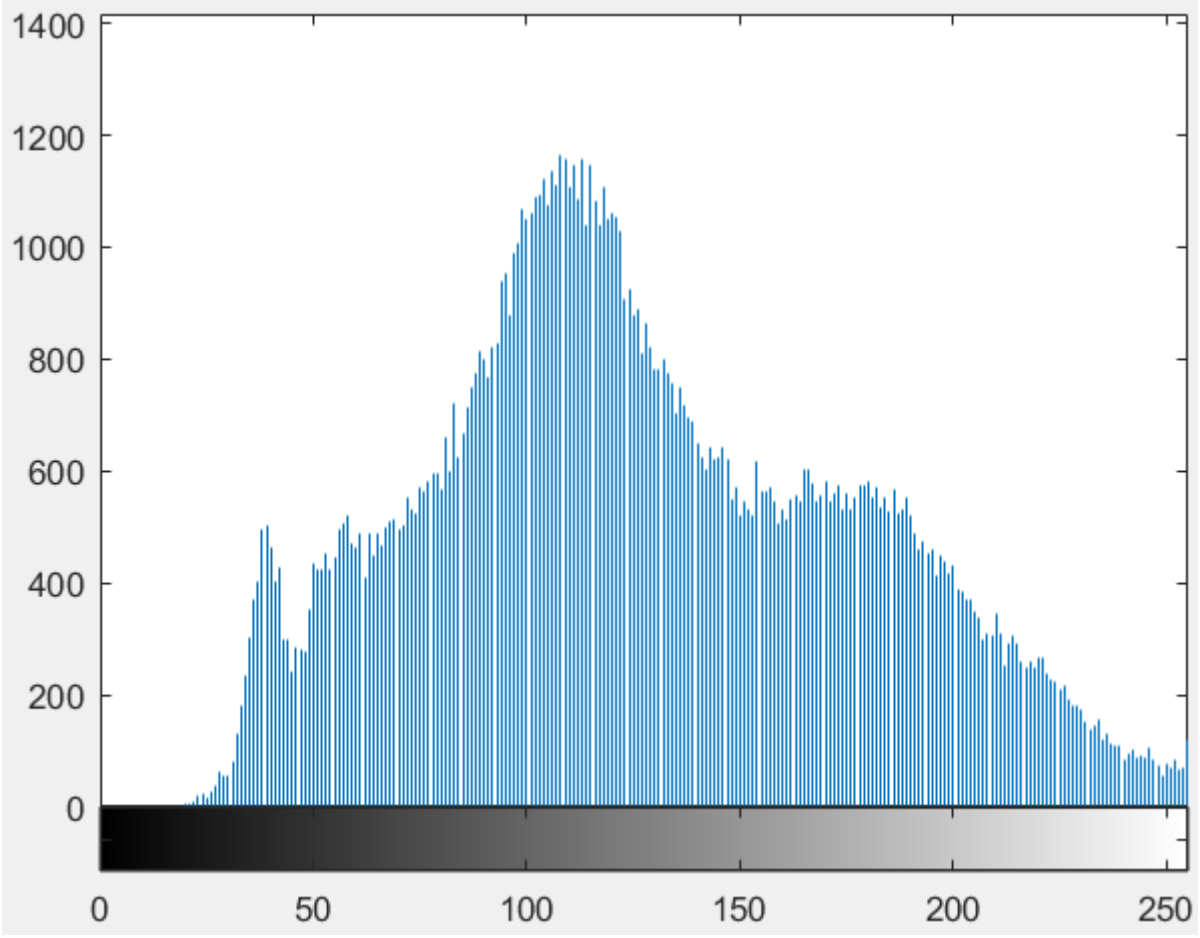
تمرین ۳

قسمت a

کدهای این قسمت در p3a_func.m و p3a_run.m قرار دارد. در فایل p3a_func تابعی نوشته‌ام که یک تصویر و یک آستانه را می‌گیرد و مقدارهای بزرگ‌تر مساوی آستانه را برابر با سفید(۲۵۵) قرار می‌دهد و مقدارهای کوچک‌تر از آستانه را برابر با سیاه (۰) قرار می‌دهد. در فایل p3a_run.m فایل kitten را خوانده‌ام سپس با سه مقدار متفاوت ۷۵، ۱۵۰ و ۲۲۵ از تصویر threshold گرفته‌ام و در انتها تصویر را ذخیره کرده‌ام و آن را نمایش داده‌ام. در زیر خروجی کد اشاره شده را مشاهده می‌کنید(برای دیدن خروجی‌ها در اندازه‌ی اصلی به پوشه‌ی p3 مراجعه کنید):

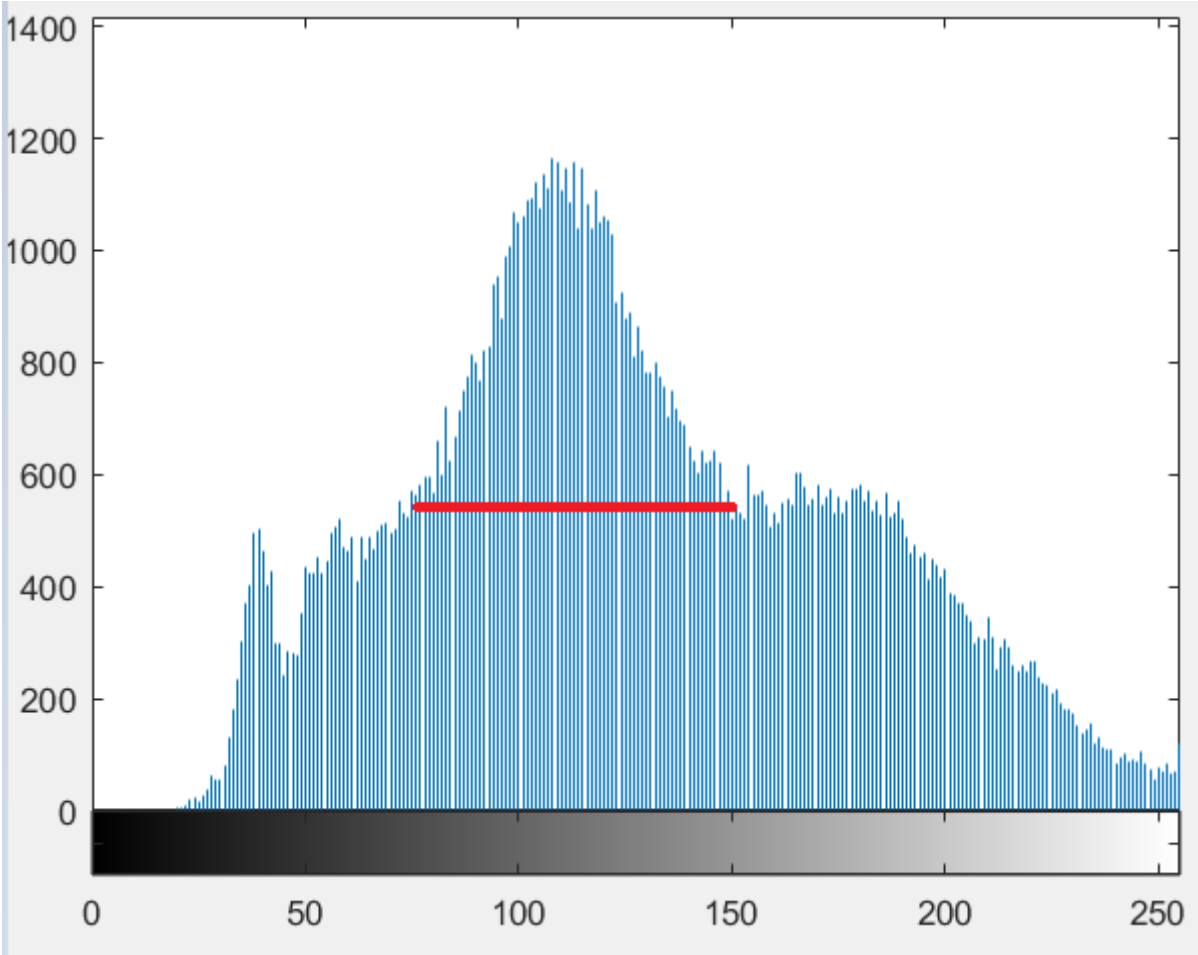
قسمت b

کد این قسمت در فایل p2b.m قرار دارد. در این قسمت از دستور imhist برای نمایش histogram تصویر گربه استفاده کرده‌ام. سپس histogram ایجاد شده را ذخیره کرده‌ام. در زیر خروجی اجرای کد را مشاهده می‌کنید:



قسمت C

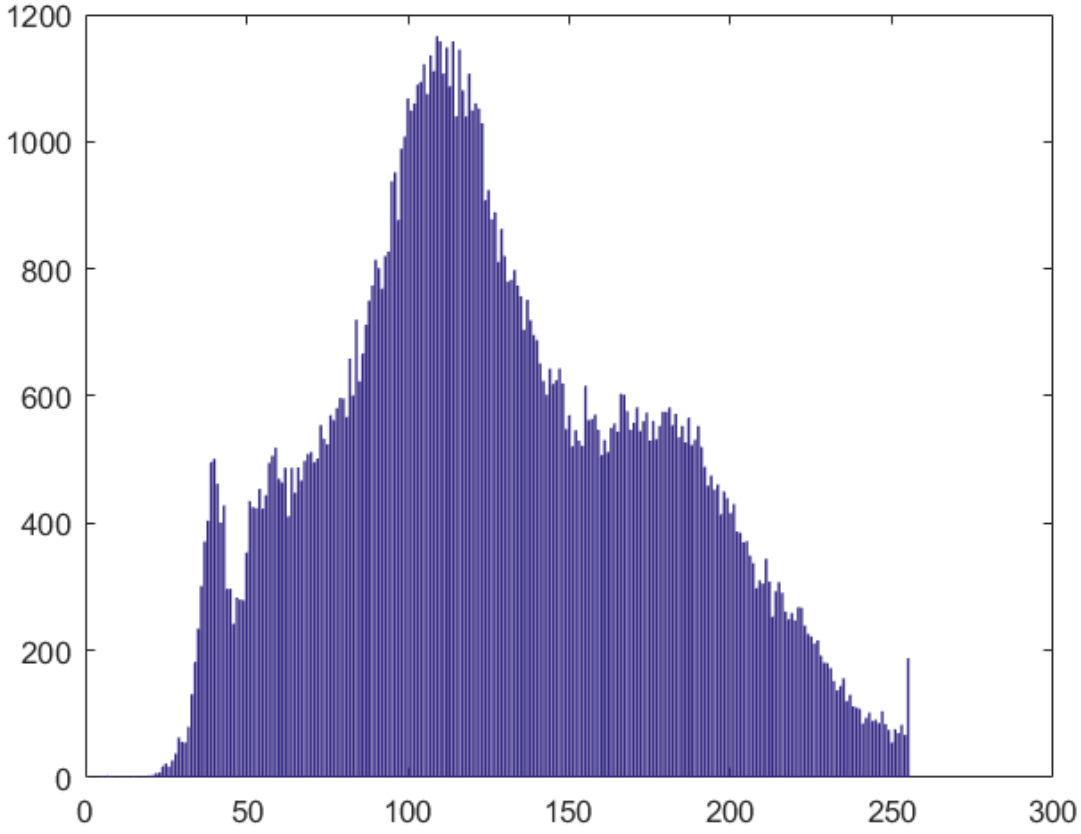
کد این سوال در p3c.m قرار دارد. ابتدا در کد Histogram را رسم کرده‌ام، همان طور که در Hostogram زیر مشاهده می‌شود،peak ها و درها های مختلفی در Histogram وجود دارد که اطلاعا ت مناسبی در مورد قسمت‌های مختلف تصویر و Object های درون آن می‌دهد.



Peakدر تصویر بالا نمایش داده شده است حاوی بخش زیادی از گربه است است به همین دلیل اگر threshold را برابر با ۱۵۰ قرار بدهیم نتیجه‌ی زیر حاصل می‌شود:

قسمت d

کدهای این بخش در فایل‌های p3d_func.m و p3d_run.m قرار دارد. پیاده‌سازی تابع مشابه imhist در فایل p3d_func.m قرار دارد. برای این کار یک ماتریس ۱*۲۵۶ ایجاد کرده‌ام و مقدار اولیه‌ی درایه‌های آن را برابر با ۰ قرار داده ام و بعد تک تک پیکسل‌های تصویر را بررسی کرده‌ام و به ازای سطح روشنایی i خانه ی i,1 ماتریس را به علاوه‌ی یک کرده‌ام بدین ترتیب تعداد سطح‌ها مختلف روشنایی را شمارده‌ام. در آخر آن را با استفاده از تابع bar رسم کرده‌ام. در زیر خروجی حاصل از کد بالا را می‌بینید که برابر با imhist است:


1200
1000
800
600
400
200
0
0
50
100
150
200
250
300

تمرین ٤

قسمت a

کد این قسمت در p4a.m قرار دارد. در این قسمت دو تصویر source و encrypted را خوانده‌ام و بعد آن‌ها را نمایش داده‌ام. در زیر خروجی این کد را مشاهده می‌کنید:

در ظاهر و با نگاه کردن تفاوتی در تصویرها دیده نمی‌شود.

قسمت b

کد این قسمت در p4b.m قرار دارد. ابتدا دو تصویر encrypted و source را خوانده‌ام سپس آن دو را با دستور isequal مقایسه کرده‌ام. خروجی isequal مشخص می‌کند دو تصویر یکسان نیستند.

قسمت c

کد این قسمت در p4c.m قرار دارد. در این قسمت دو تصویر encrypted و source را با عملکرد == مقایسه کرده‌ام که خروجی آن یک تصویر logical است که تصویر جدید در پیکسل‌هایی که دو تصویر یکسان بوده‌اند ۱ (سفید) است و در سایر نقاط ۰(سیاه) است. در زیر پیام مخفی موجود در تصویر را مشاهده می‌کنید:

قسمت d

کد این قسمت در p4d.m قرار دارد. در این قسمت هم مانند قسمت c عمل کرده‌ام و از عملکرد منطقی == برای پیدا کردن تفاوت دو تصویر استفاده کرده‌ام، تصویر جدید در پیکسل‌هایی که دو تصویر یکسان بوده‌اند ۱ (سفید) است و در سایر نقاط ۰(سیاه) است. در زیر پیام مخفی موجود در تصویر را مشاهده می‌کنید:

قسمت e

کدهای این قسمت در p4e.m قرار دارد. اینجا هم مانند دو قسمت قبل عمل کردم و از عملکرد == استفاده کردم. در زیر تصویر خروجی این کد را می‌بینید:

تمرین ۵

قسمت a

کد این قسمت در فایل p5a.m قرار دارد. این سوال را به صورت حل کرده‌ام، در روش اول تصویر اصلی و فیک مانالیزا را ازRGB به Gray Scale تبدیل کرده‌ام و سپس با استفاده از عملکرد == تفاوت دو تصویر را پیدا کرده‌ام. در روش دوم دو تصویر RGB را از هم کم کرده‌ام و بعد از stretching آن را نمایش داده‌ام. در پایان کد هم دو تصویر را ذخیره کرده‌ام. در زیر دو خروجی کد را مشاهده می‌کنید(برای دیدن تصویرها در سایز واقعی به پوشه‌ی p5a مراجعه کنید):

روش اول:

روش دوم:

همین‌طور که در تصویر سمت راست پایین مشاهده می‌شود، نام کپی کننده‌ی تصویر Ali Abbasi است.

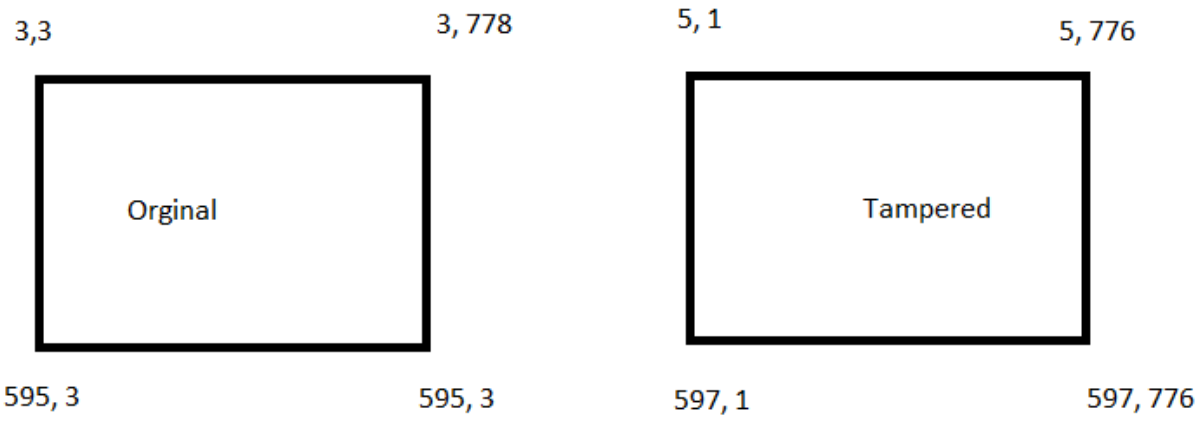
قسمت b

کدهای این قسمت در فایل p5b.m قرار دارد. در این قسمت باید برای دو دسته تصویر Tempering Detection انجام بدیم.

تصویر Irises :

دو تصویر اصلی دستکاری شده‌ی Irises را در زیر می‌بینید.

همین‌طور که مشاهده می‌شود در دو تصویر، المنت‌های مختلف در مکان یکسانی قرار ندارند به همین دلیل نمی‌توان با عملکرد == یا منها کردن دو تصویر تفاوت‌ها را به خوبی پیدا کرد به همین دلیل دو تصویر را در کد ابتدا بر اساس جزییات زیر برش داده‌ام:



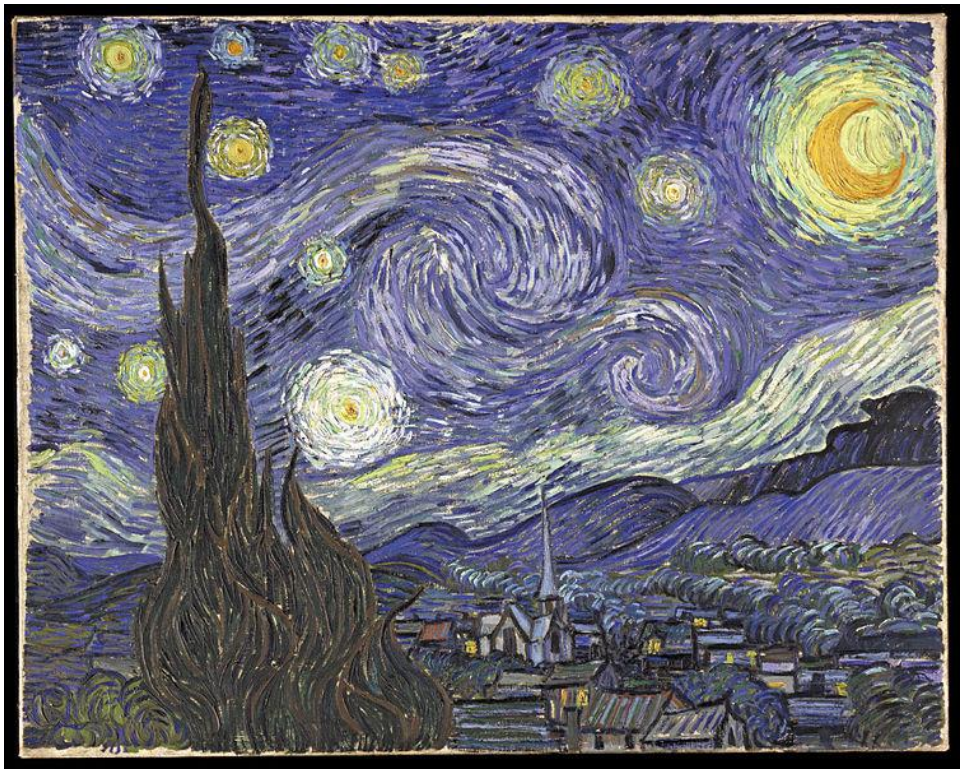
بعد از این کار قسمت‌های مختلف تصویر نسبت به هم در محل یکسانی قرار می‌گیرند. سپس دو تصویر جدید را از هم کم کرده‌ام و بعد با تابع به توان ۳ تصویر حاصل را stretch کرده‌ام. سپس آن را به تصویر Black and White تبدیل کرده‌ام. در زیر خروجی این قسمت از کد را می‌بینید:

همین‌طور که مشاهده می‌شود دو تصویر در نقاط سفید بسیار متفاوت بوده‌اند.

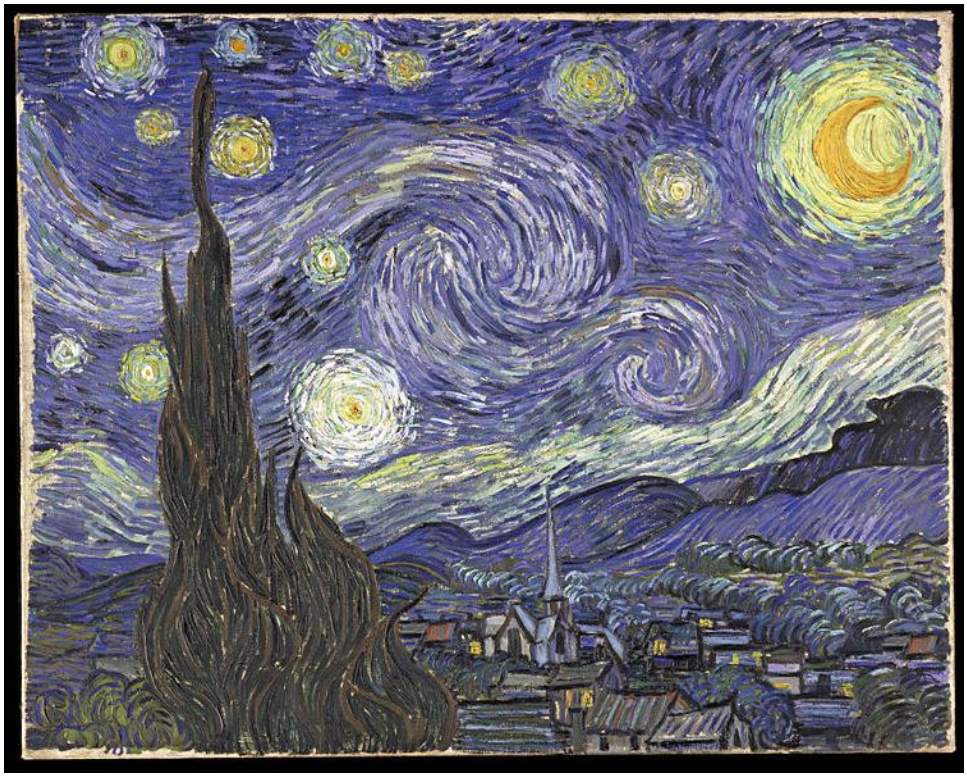
تصویر شب ستاره‌دار:

کدهای این قسمت در p5b.m قرار دارد. در زیر دو تصویر ورودی را مشاهده می‌کنید.

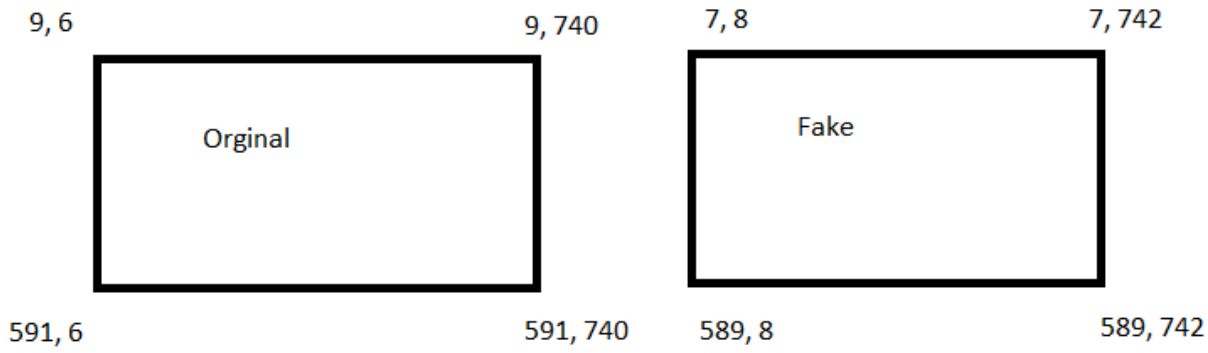
تصویر اصلی:



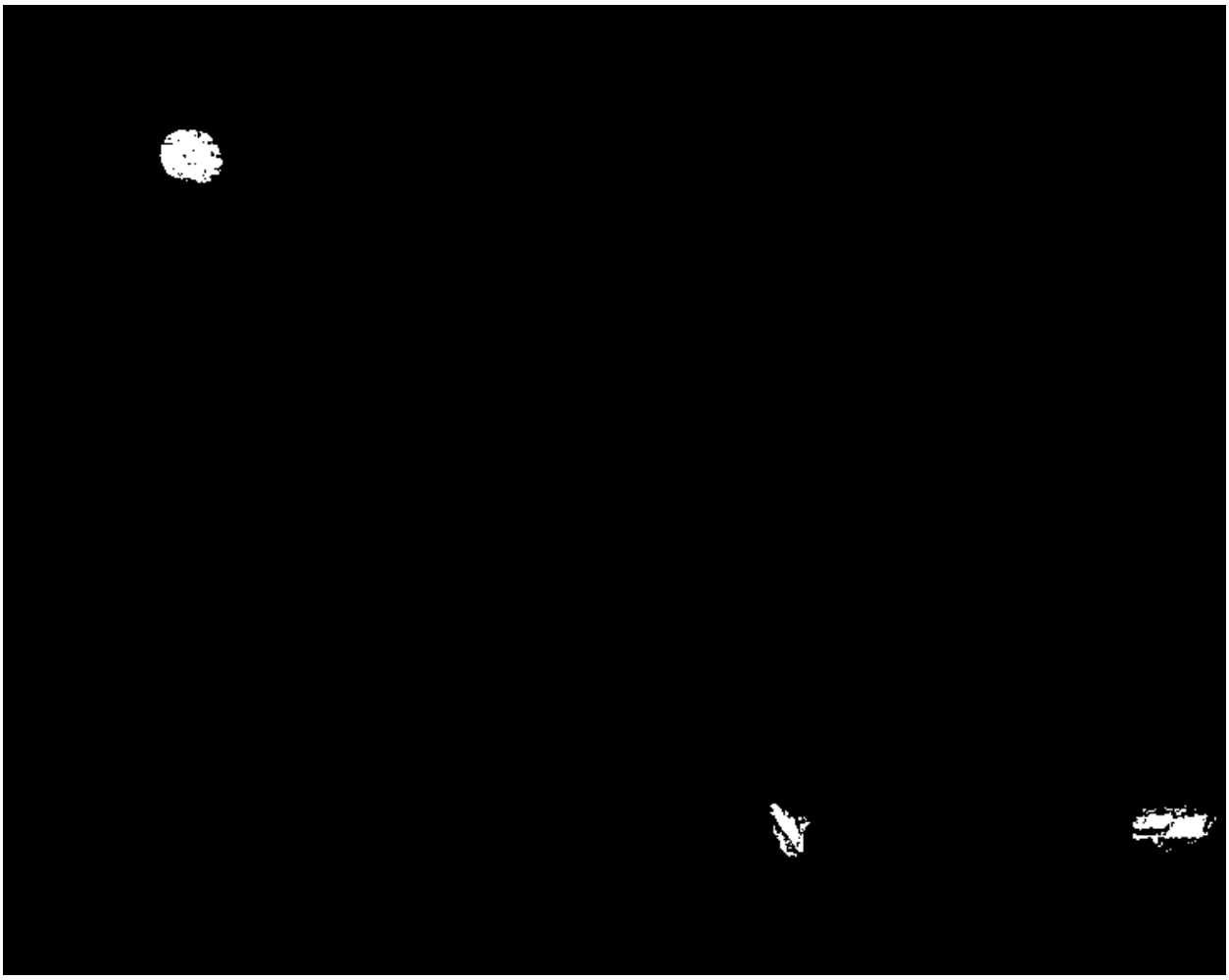
تصویر جعلی:



در این قسمت حاشیه‌های سیاه رنگی با اندازه‌های مختلف به تصویر نقاشی‌ها اضافه شده است، طبق جزییات زیر دو تصویر را برش می‌دهیم تا قسمت‌های یکسان دو تصویر نسبت به هم در یک پیکسل قرار بگیرند.



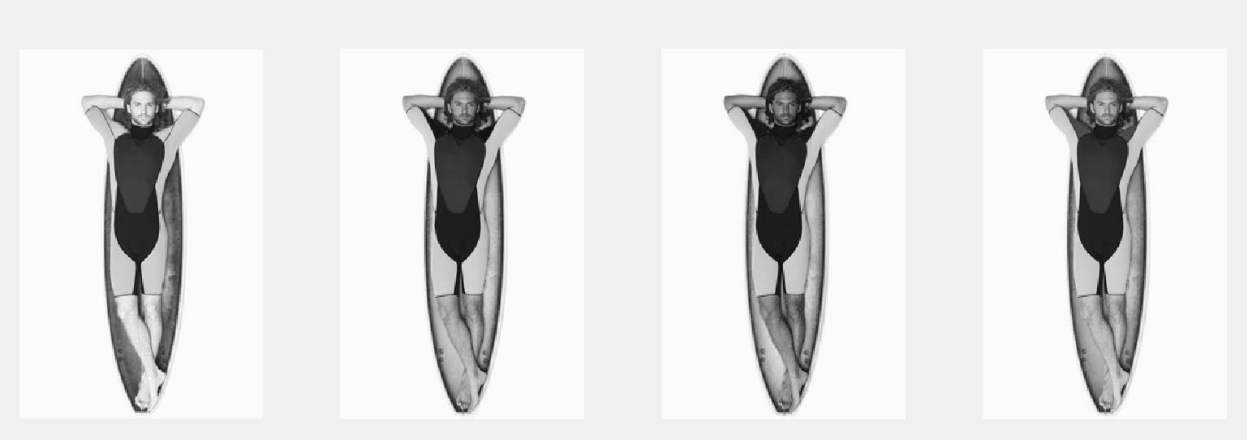
بعد از این کار قسمت‌های مختلف تصویر نسبت به هم در محل یکسانی قرار می‌گیرند. سپس دو تصویر جدید را از هم کم کرده‌ام و بعد با تابع به توان ۳ تصویر حاصل را stretch کرده‌ام. سپس آن را به تصویر Black and White تبدیل کرده‌ام. در زیر خروجی این قسمت از کد را می‌بینید:

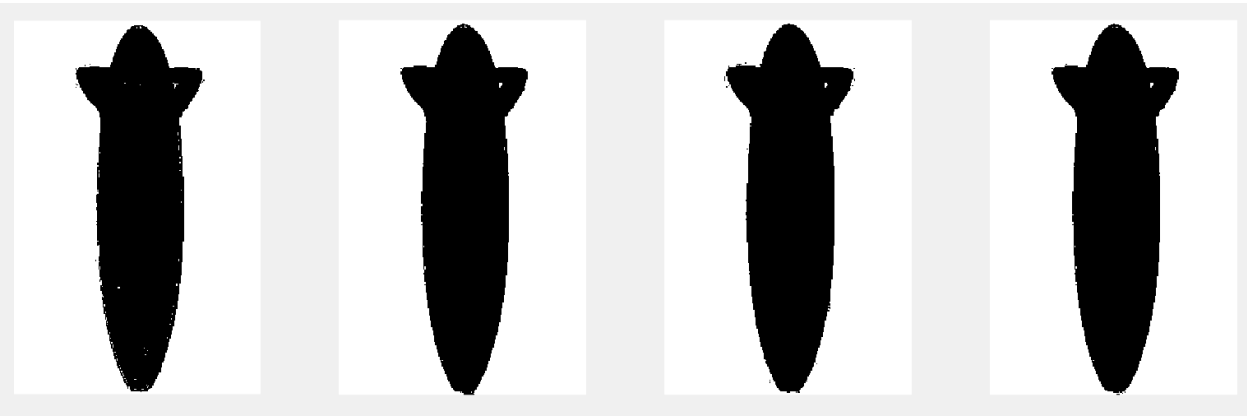


در قسمت‌های سفید دو تصویر متفاوت‌اند و در قسمت‌های سیاه رنگ تفاوتی دیده نمی‌شود.

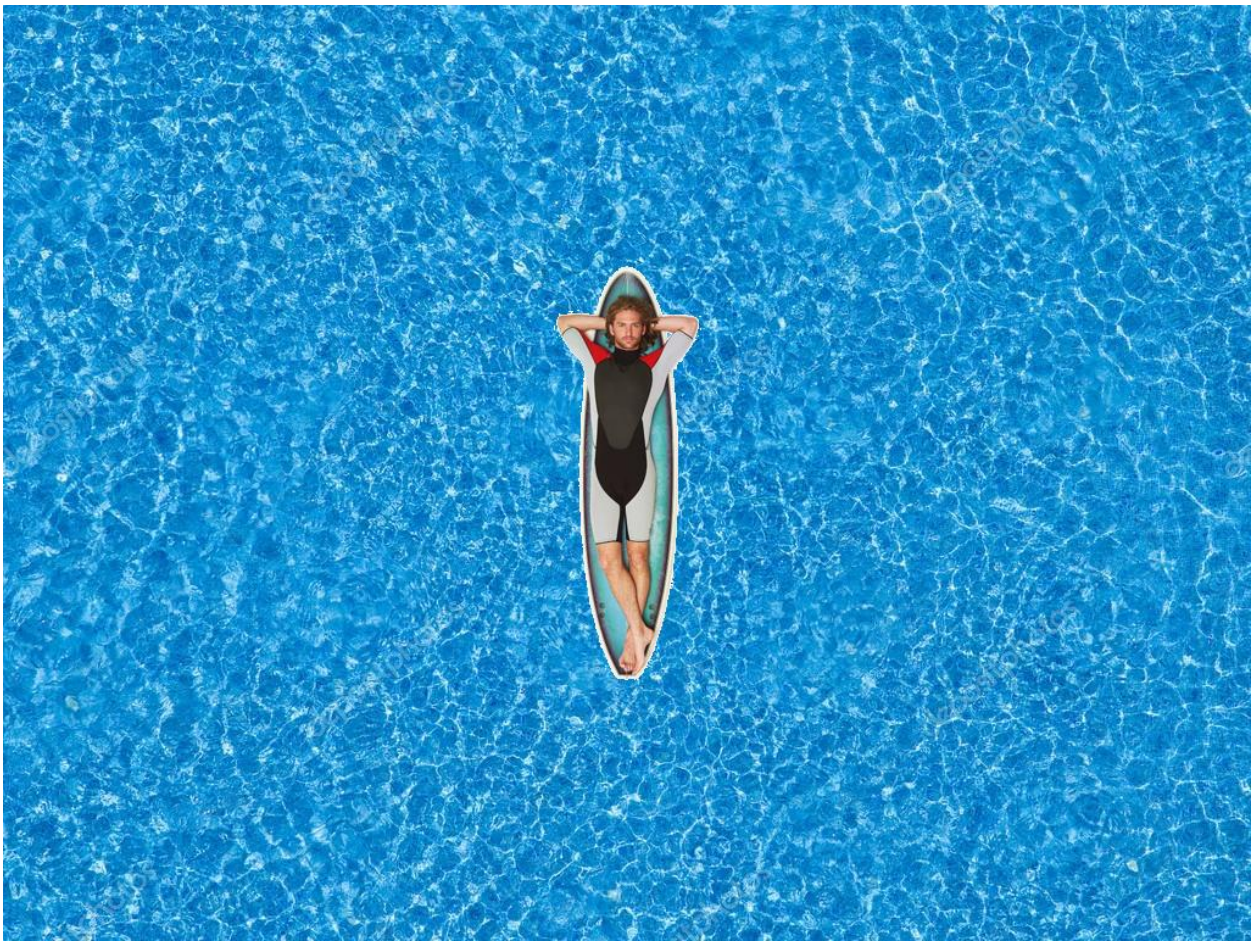
تمرین ٦

کد این قسمت در p6.m قرار دارد. برای این کار ابتدا تصویر مرد شناور را به درصد اندازه‌ی اولیه خود رسانده‌ام تا وقتی آن را بر روی تصویر دریا قرار دادیم واقعی تر به نظر برسد. برای قرار دادن تصویر مرد شناور به یک Mask نیاز داشتم که در محل‌هایی که ماسک برابر یک است از تصویر دریا استفاده کنم و در محل‌هایی که ماسک برابر با است از تصویر مرد شناور استفاده کنم به همین دلیل تصویر مرد شناور در طیف قرمز، سبز، آبی و gray scale را رسم کردم و این چهار تصویر را به تصویر logical تبدیل کردم تا یک Mask بسازم. در زیر چهار Mask را مشاهده می‌کنید:





همین‌طور که مشاهده می‌شود Mask مرد شناور در طیف آبی از سایرین بهتر است در نتیجه از آن استفاده می‌کنیم. در محل‌هایی که ماسک سیاه است از مرد شناور استفاده می‌کنیم و در سایر محل‌ها از تصویر دریا. خروجی کد را در تصویر زیر مشاهده می‌کنید، برای دیدن تصویر در اندازه‌ی واقعی به پوشه‌ی p6.m مراجعه کنید.

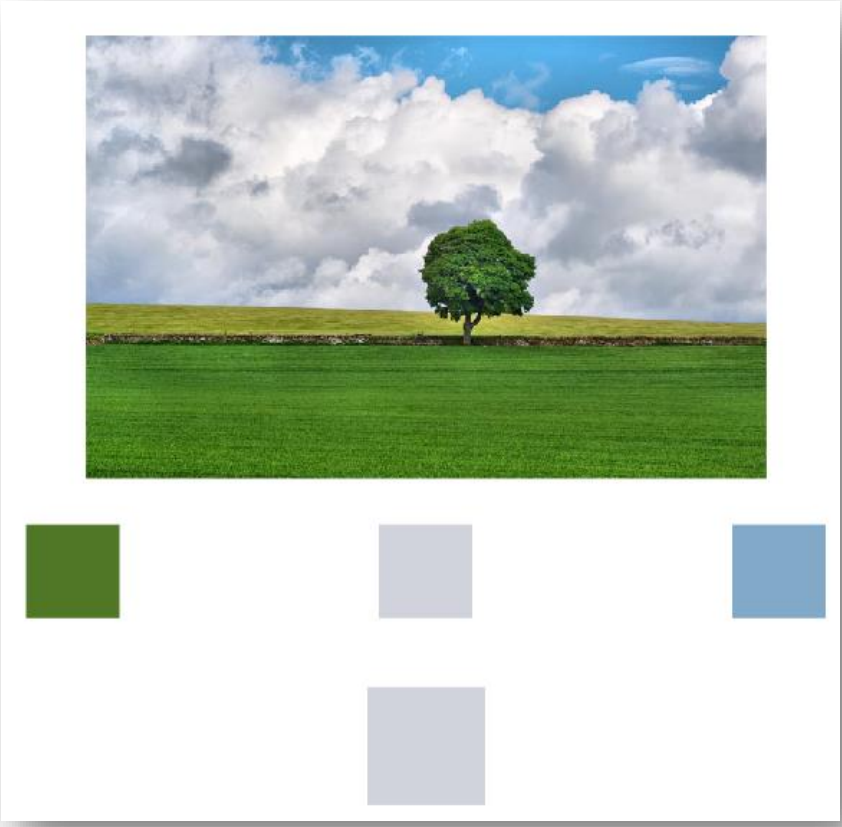


تمرین ۷

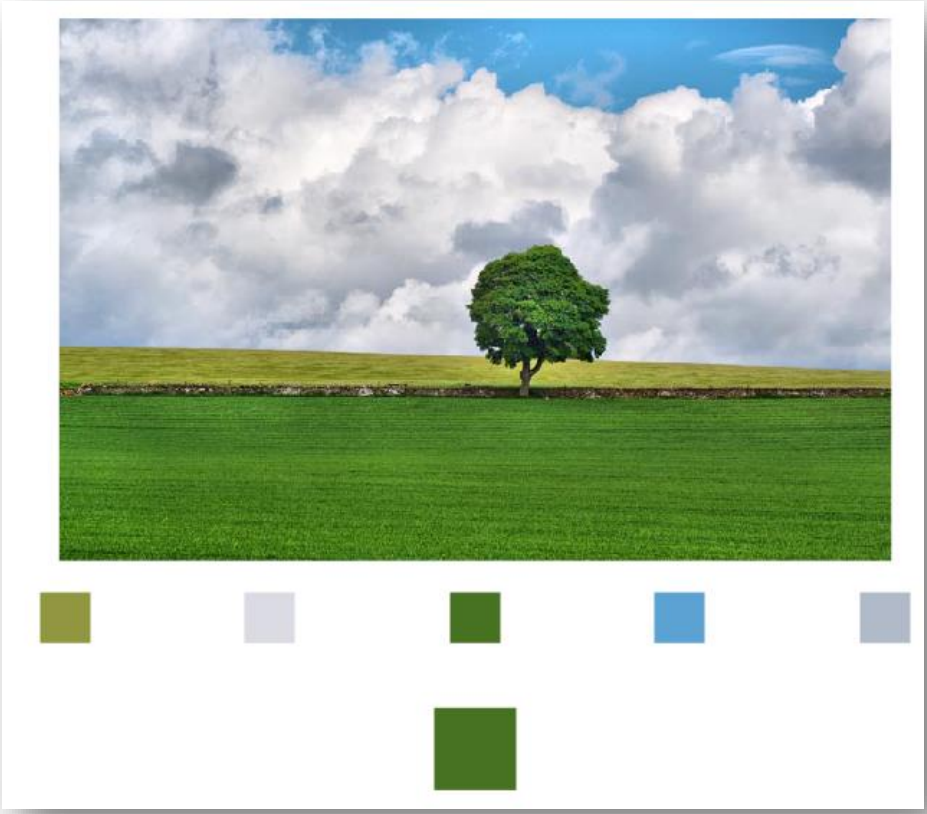
قسمت a

کد این قسمت در فایل‌های p7func.m و p7a_run.m قرار دارد. در فایل p7a_run.m تابعی قرار دارد که palette یک تصویر ورودی را به ازای تعداد رنگ‌ها مشخص شده که باید در palette باشند را پیدا می‌کند. این تابع از الگوریتم kmeans برای این کار استفاده می‌کند. تصویر را به ماتریسی دو بعدی تبدیل می‌کنیم که هر سطر آن حاوی سه عدد است که آن سه عدد RGB یک پیکسل هستند و به تعداد پیکسل‌های تصویر آن ماتریس سطر دارد. این ماتریس را به تابع kmeans متلب می‌دهم و کلاستری که هر سطر از ماتریس در آن قرار می‌گیرد و centroid های Kmeans را دریافت می‌کنم. سپس بیشتر کلاستری را که تکرار شده است را پیدا می‌کنم. در نهایت centroidها و Dominan Color را برمی‌گردانم. در p7□□□□□.□ هم تابع را به ازای سایزهای مختلف palette فراخوانده‌ام و آن را رسم کرده‌ام. در زیر خروجی کد را مشاهده می‌کنید(**برای دیدن تصاویر در اندازه‌ی واقعی** به پوشه‌ی p7 مراجعه کنید):

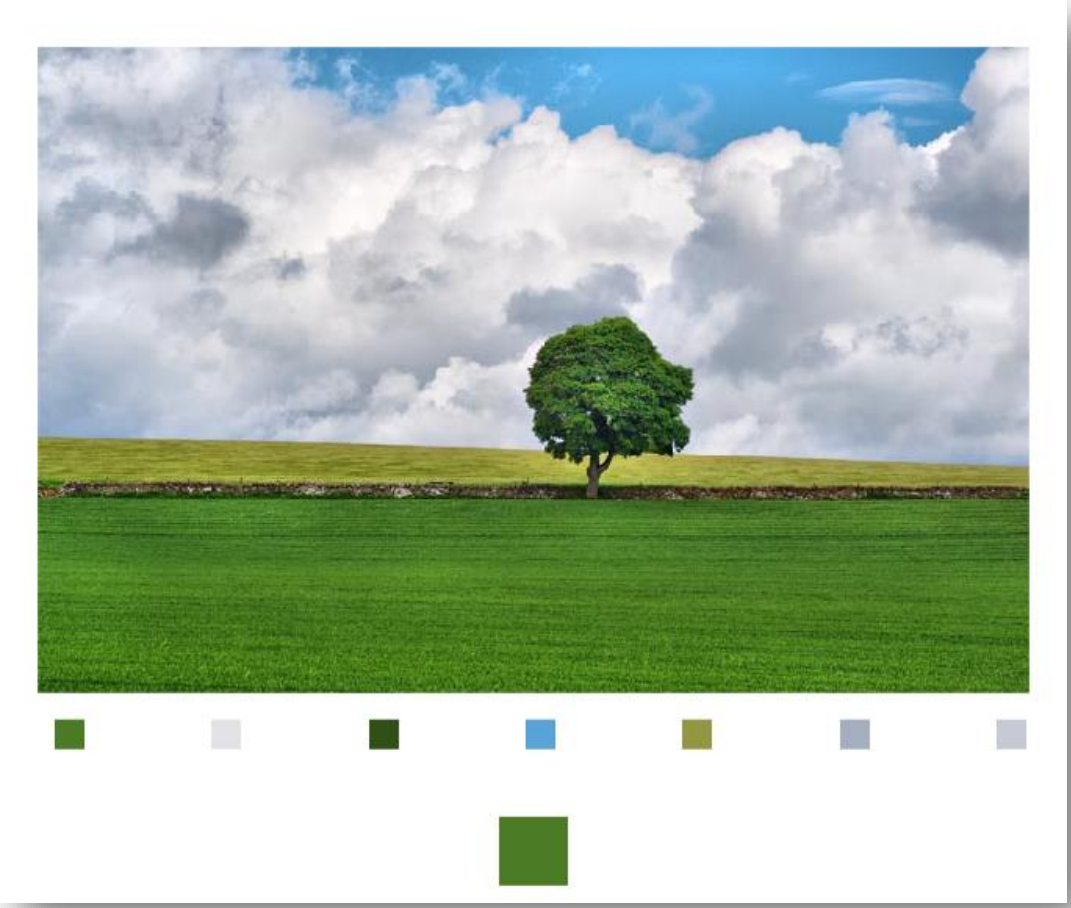
سایز palette برابر ۳(سطر اول تصویر ورودی، سطر دوم palette، سطر سوم Dominant Color):



سایز palette برابر ۵(سطر اول تصویر ورودی، سطر دوم palette، سطر سوم Dominant Color):



سایز palette برابر ۷(سطر اول تصویر ورودی، سطر دوم palette، سطر سوم Dominant Color):

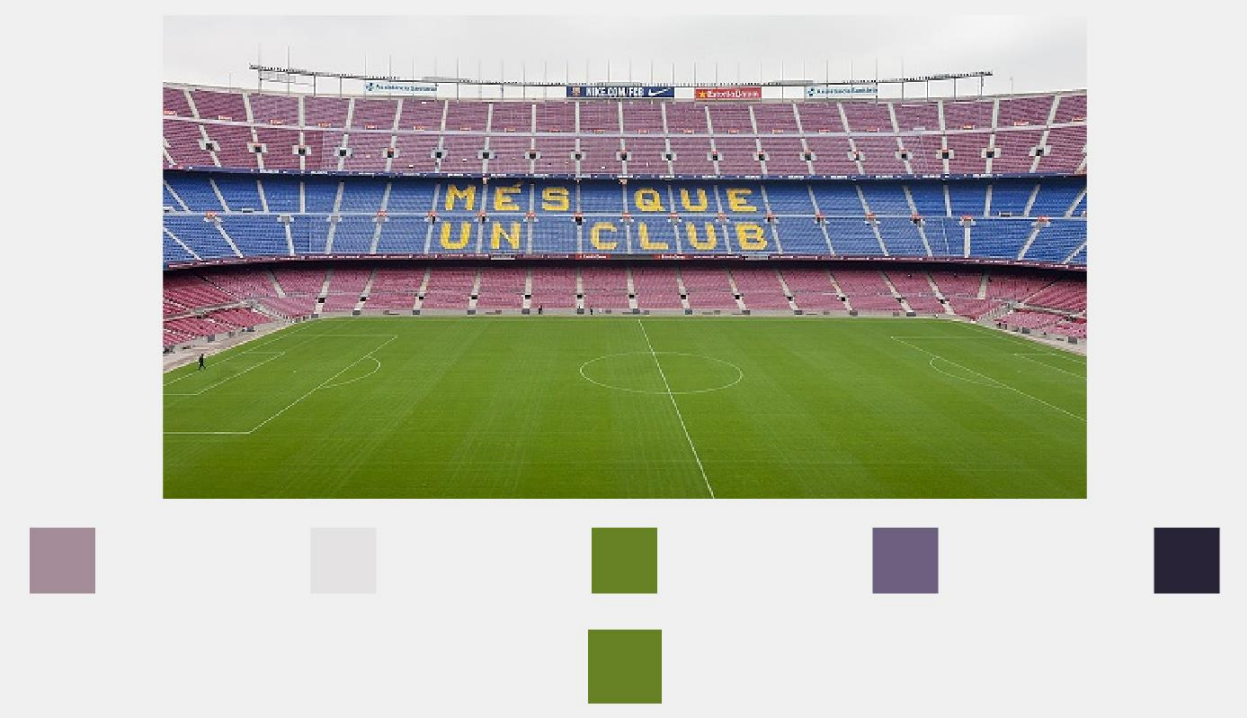


قسمت b

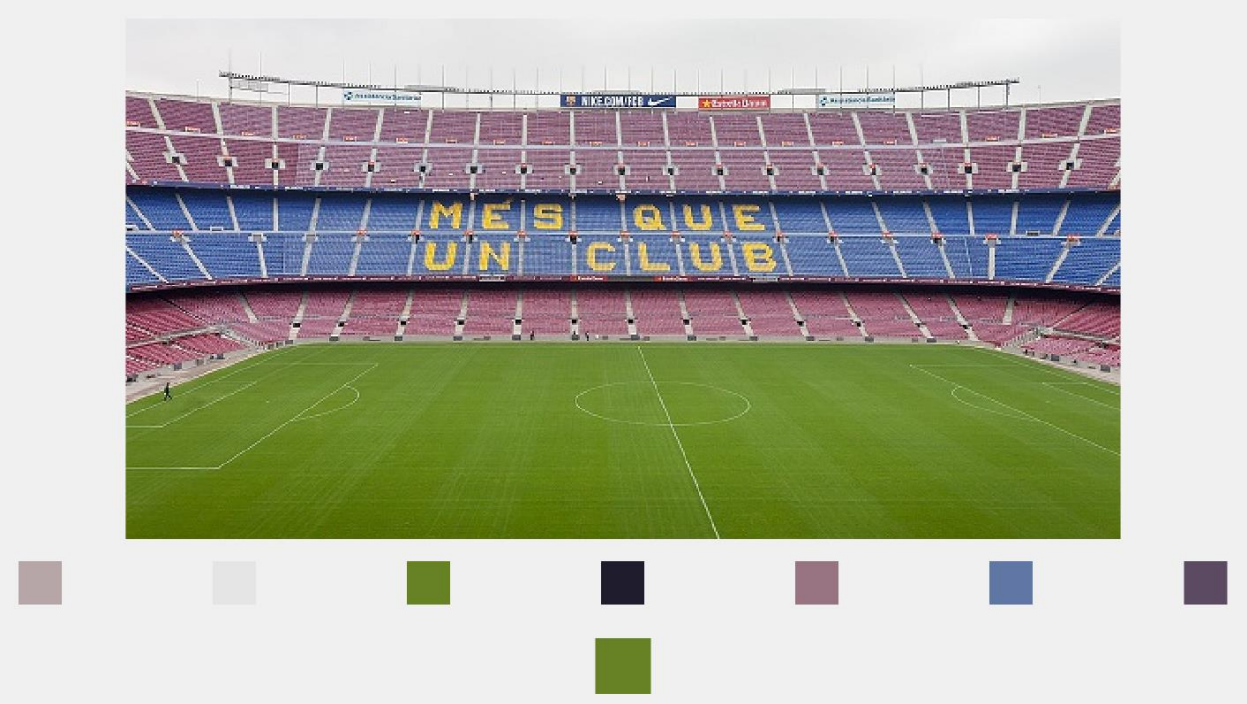
کد این قسمت در فایل‌های p7func.m و p7b_run.m قرار دارد. این قسمت دقیقا مانند قسمت قبل است با این تفاوت که از تصویر Noucamp برای ورودی استفاده می‌کنیم و سایز palette متفاوت است. به همین دلیل کد را دوباره توضیح نمی‌دهم.

در زیر خروجی کد را مشاهده می‌کنید(**برای دیدن تصاویر در اندازه‌ی** واقعی به پوشه‌ی p7 مراجعه کنید):

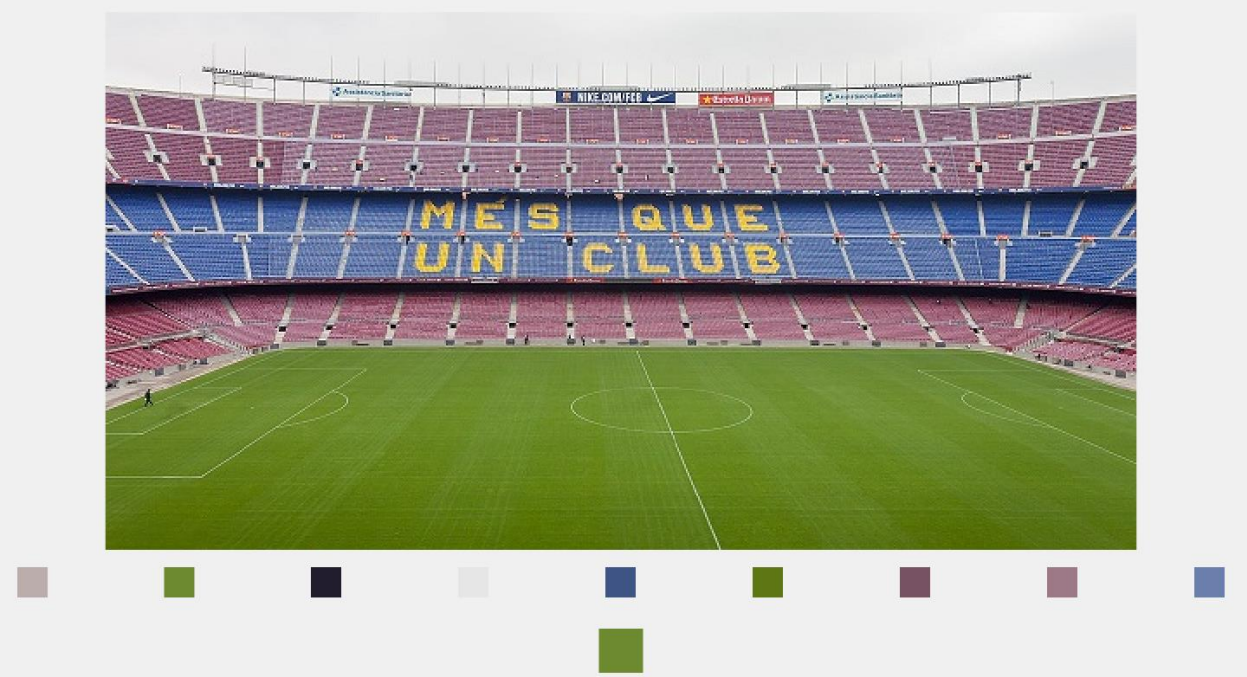
سایز palette برابر۵(سطر اول تصویر ورودی، سطر دوم palette، سطر سوم Dominant Color):



سایز palette برابر ۷(سطر اول تصویر ورودی، سطر دوم palette، سطر سوم Dominant Color):



سایز palette برابر۹(سطر اول تصویر ورودی، سطر دوم palette، سطر سوم Dominant Color):



قسمت c

کدهای این قسمت در p7_func.m و p7c_run.m قرار دارد. در p7_func.m از همان کد پیدا کردن palette در قسمت a و b این سوال استفاده شده که با استفاده از الگوریتم کلاسترینگ kmeans یک palette برای تصویر ورودی می‌ساخت. آن کد را کمی تغییر داده‌ام و بعد فراخوانی kmeans و پیدا کردم کلاسترهای تصویر و کلاستری که هر پیکسل در آن قرار می‌گیرد، تصویری جدید هم سایز تصویر ورودی ساخته‌ام و هر پیکسل را با کلاستری (palette color) که در آن قرار گرفته‌است پر کرده‌ام. در p7c_run.m تابع بالا را برای سایزهای مختلف palette و برای دو تصویر nature و noucamp فراخوانی کرده‌ام.

در زیر خروجی‌های کد بالا را مشاهده می‌کنید ، برای دیدن تصاویر در اندازه واقعی به پوشه‌ی p7 مراجعه کنید.

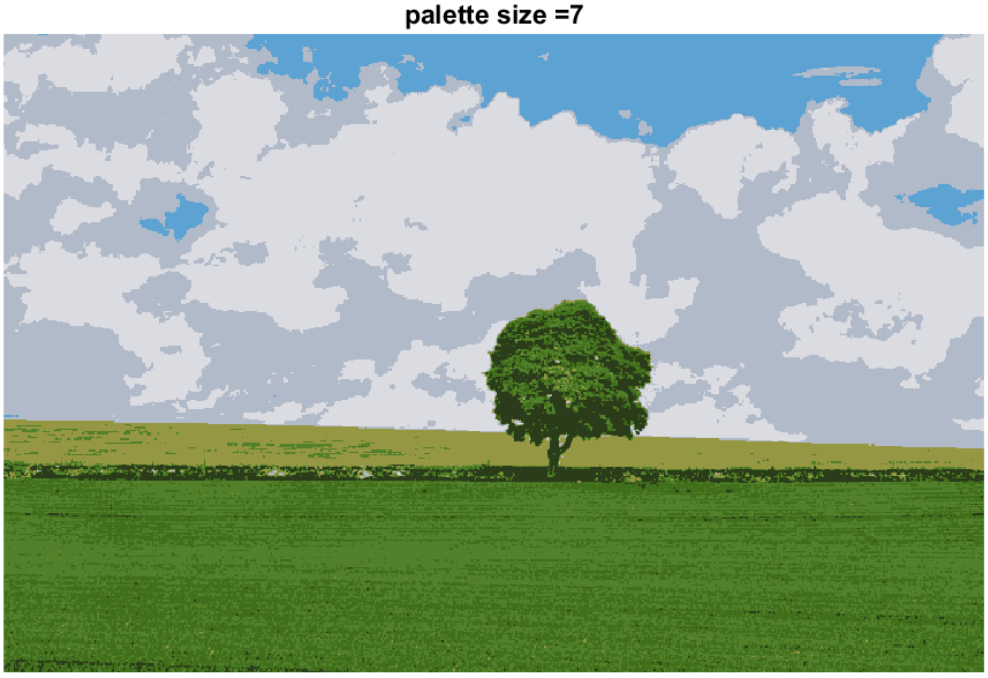
سایز palette برابر ۳ :

palette size =3

سایز palette برابر ۵ :

palette size =5

سایز palette برابر ۷ :



سایز palette برابر ۵ :

سایز palette برابر ۷ :

palette size =7

سایز palette برابر ۹ :

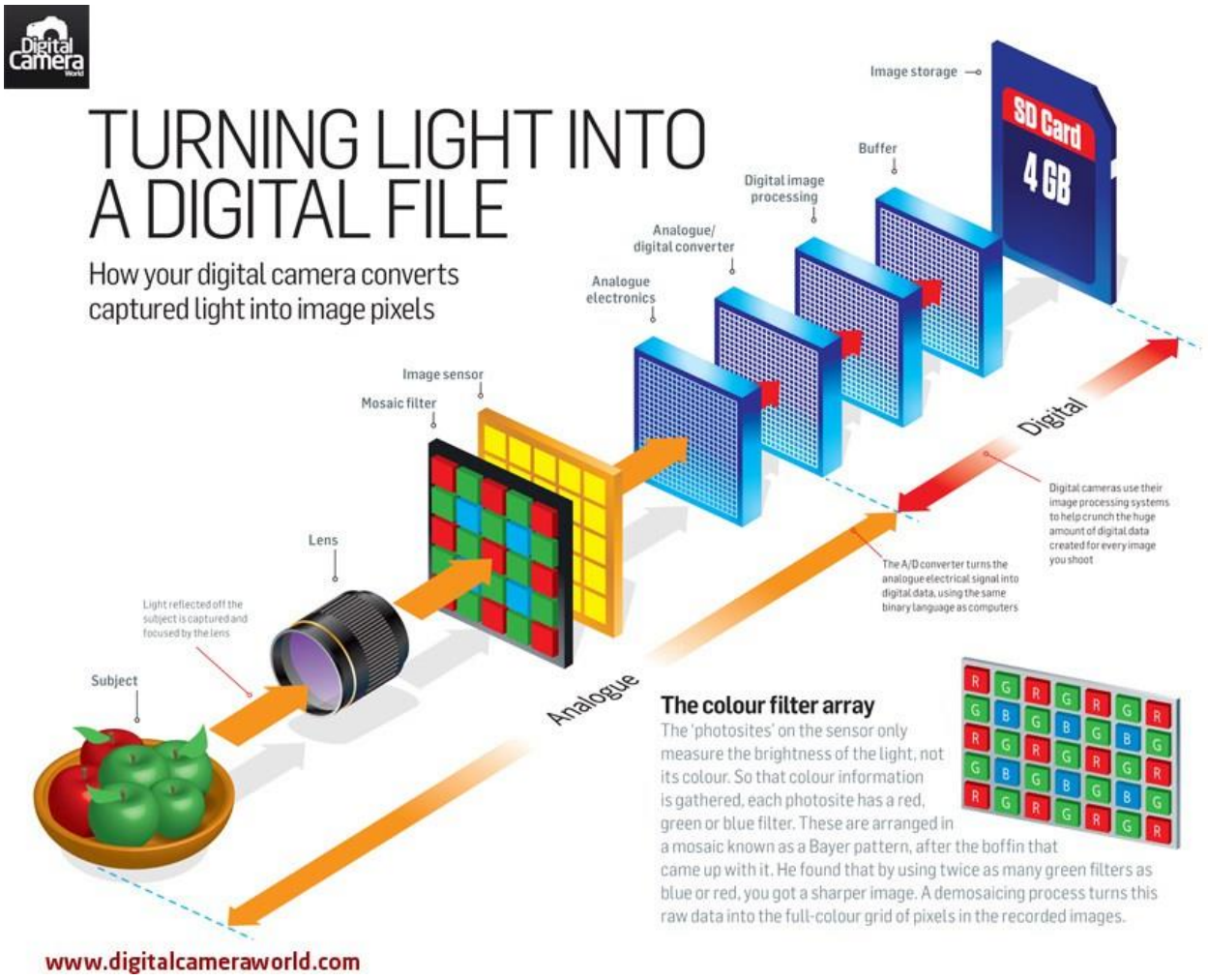
palette size =9

تمرین ۸

قسمت a

لنز جلوی دهانه‌ی دوربین فوتون‌های نور متمرکز می‌کند تا بتوان در مراحل بعد یک تصویر شفاف از یک جسم با فاصله‌ی مشخص از جلوی دوربین گرفت. با تنظیم محل لنز می‌توان تنظیم کرد که روی چه قسمت‌هایی از جلوی دوربین تمرکز کرد و روی چه قسمت‌هایی کمتر فوکوس کرد. زمانی که دکمه‌ی گرفتن تصویر را می‌زنیم شاتر به کنار می‌رود و به نور اجازه می‌دهد به سنسور بتابد. در دوربین‌های دیجیتال فیلمی وجود ندارد بلکه یک قطعه‌ی الکترونیکی است که پرتوهای ورودی که به آن می‌تابد را می‌گیرد و آن را به سیگنال‌های الکترونیکی تبدیل می‌کند. این نوع سنسورها معمولا دو نوع‌اند: charge-coupled device و CMOS Image Sensor . سنسور دوربین از هزاران قسمت کوچک تشکیل شده است که تصویر ورودی (نورهای ورودی) با برخورد به آن‌ها صورت تعدادی زیادی قسمت تقسیم می‌شود که به آن پیکسل می‌گویند. هر پیکسل از سنسور رنگ و روشنایی قسمتی از تصویر ورودی را می‌سنجد و آن را به صورت یک عدد در می‌آورد. هر عکس دیجیتال دنباله‌ای از عددها است.

در تصویر زیر خلاصه‌ای از این موضوع را مشاهده می‌کنید:



قسمت b

Image Quality یک ویژگی تصویر است که میزان تغییرات و تفاوت تصویر دریافت شده از یک صحنه نسبت به تصویر ایده‌آل را بیان می‌کند. یک تصویر از یک صحنه به صورت شیمیایی و یا الکترونیکی در دوربین ایجاد می‌شود، در سیستم دوربین‌های فعلی نور از دریچه‌ی دوربین عبور می‌کند و بر روی فیلم/سنسور درون دوربین می‌افتد و تصویر ثبت می‌شود. تصویر ایجاد شده در این فرآیند یک تقریب از صحنه و تصویر ایده آل است. در واقع Image Quality بیان می‌کند چه مقدار تصویر تقریب زده شده از صحنه به تصویر ایده‌آل نزدیک است. علاوه بر فرآیند ثبت تصویر انتقال و ذخیره سازی آن هم باعث تغییر میزان شباهت تصویر تقریب زده شده از صحنه می‌شود. وقتی می‌گوییم یک تصویر Quality بهتری نسبت به یک تصویر دیگر دارد منظور این است آن تصویر تقریب نزدیک‌تری به تصویر ایده‌آل از صحنه است. برای سنجیدن کیفیت یک تصویر روش‌های عدد و غیر

عددی مختلف وجود دارد. برای روش‌های غیر عددی به عنوان مثال می‌توان از انسان کمک گرفت. برای روش‌های عددی راه‌های مختلفی وجود دارد که یکی از آن راه ها تعریف ویژگی‌های مختلفی برای تصویر است مانند:

Noise, Contrast, Distortion, Sharpness, Artifacts, Lens flare, Vignetting

قسمت C

تعداد پیکسل‌های یک تصویر مشخص می‌کند یک تصویر از چه تعداد پیکسل تشکیل شده است و الزاما زیاد بودن تعداد پیکسل‌ها دلیل بر بالا بودن کیفیت تصویر ندارد زیرا کیفیت تصویر به لنز، سنسور، نحوه‌ی ذخیره سازی و ... هم بستگی دارد. همچنین تعداد پیکسل بالا بر روی یک سنسور کوچک باعث زیاد شدن نویز می‌شود.

قسمت D

تعداد بیت‌های یک پیکسل تعداد رنگ‌هایی که آن پیکسل می‌تواند نشان دهد را مشخص می‌کند. اگر n بیت باشد هر پیکسل ۲^n رنگ متفاوت را می‌تواند نشان بدهد. در صورتی که دوربین هر دو تصویر کاملا از همه‌ی لحاظ شبیه هم باشند و فقط در تعداد بیت‌های رنگ متفاوت باشند تصویری که بیت‌های بیشتری دارد کیفیت بهتری دارد زیرا تقریب بهتری از صحنه (تصویر ایده‌آل) است.