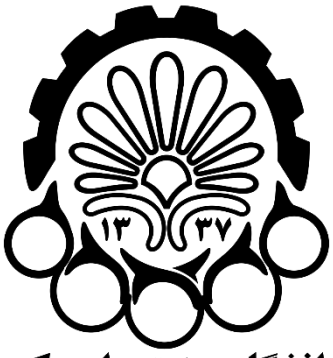
به نام خدا



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

تمرین درس پردازش تصویر–سری اول

فردین آیار

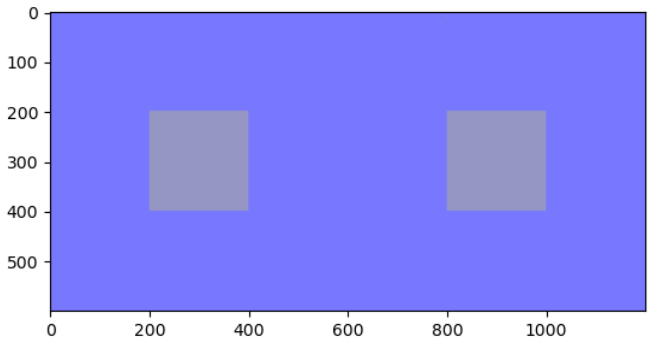
شماره دانشجویی: ۹۹۱۳۱۰۴۰

استاد: دکتر رحمتی

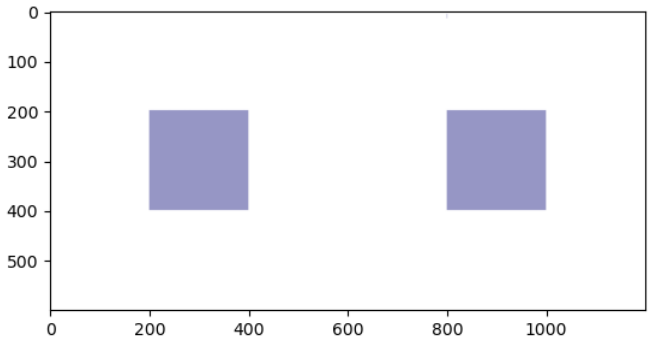
دانشکده کامپیوتر– زمستان ۹۹

۱)

a) کد مربوط به این بخش در فایل a.py قرار دارد. با استفاده از یک دستور ساده، پس‌زمینه سمت چپ تصویر را هم‌رنگ سمت راست تصویر می‌کنیم. خروجی در شکل۱ نشان داده شده است. همانطور که مشاهده می‌شود مربع‌های کوچک درون شکل هم‌رنگ هستند. به نظر می‌رسد در شکل اصلی، تضاد رنگ مربع‌ها با پس‌زمینه سبب متفاوت به نظر رسیدن رنگ آن‌ها می‌شود. برای مشخص شدن رنگ واقعی مربع‌ها، در شکل۲ پس‌زمینه را به طور کامل حذف می‌کنیم.

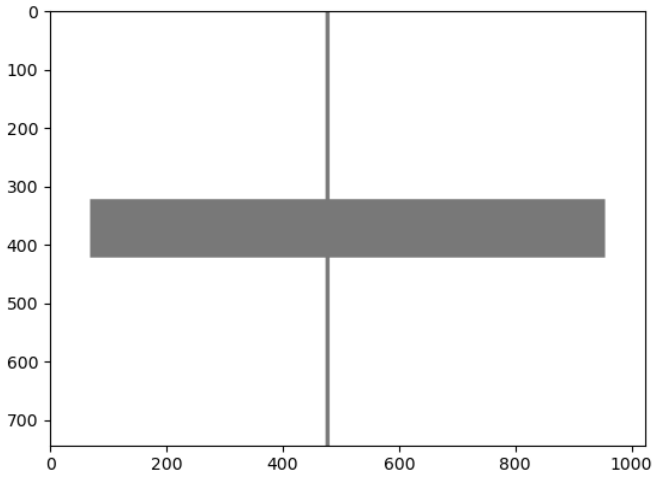


شکل ۱



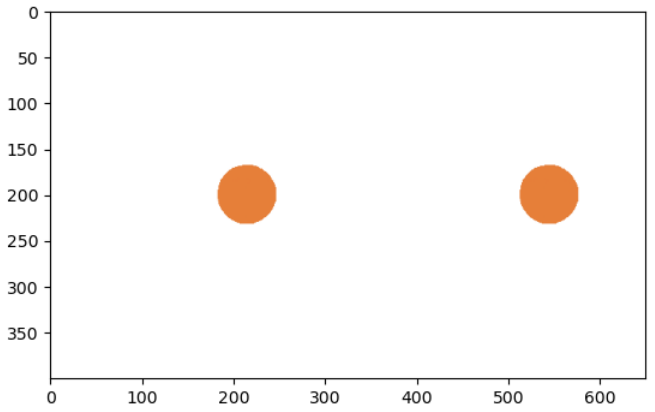
شکل ۲

b) کد مربوط به این بخش در فایل b.py قرار دارد. ابتدا مقدار پیکسل وسط تصویر را بدست می‌آوریم. سپس تمام پیکسل‌هایی را که مقدار آن‌ها با پیکسل وسط یکسان نیست، سفید می‌کنیم. خروجی در شکل۳ نمایش داده شده است. همانطور که مشاهده می‌شود مستطیل درون شکل کاملا یک رنگ است.



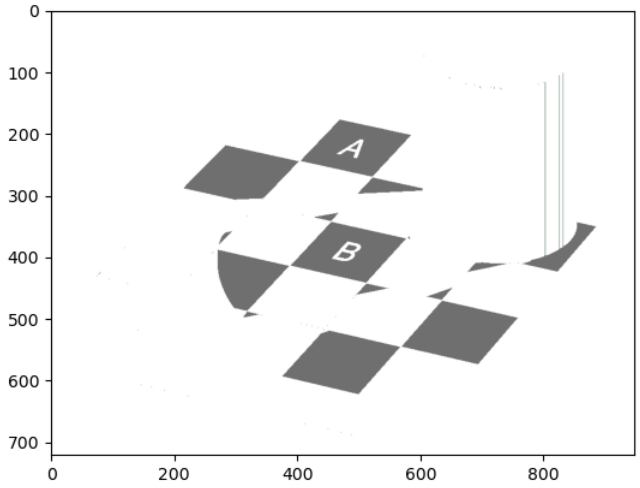
شکل ۳

c) کد مربوط به این بخش در فایل c.py قرار دارد. این تصویر در فرمت RGBA است. بعد از حذف کانال A، رنگ دایره‌های نارنجی را بدست می‌آوریم. سپس همه پیکسل‌هایی که رنگ آن‌ها با دایره‌های نارنجی یکسان نیست، سفید می‌کنیم. مطابق شکل۴ دایره‌های نارنجی هم‌اندازه هست. توجه شود مرز دایره‌ها از حالت نرم خارج شده است؛ زیرا برخی پیکسل‌های مرزی دارای رنگ اندکی متفاوت بوده و حذف شده‌اند.



شکل ۴

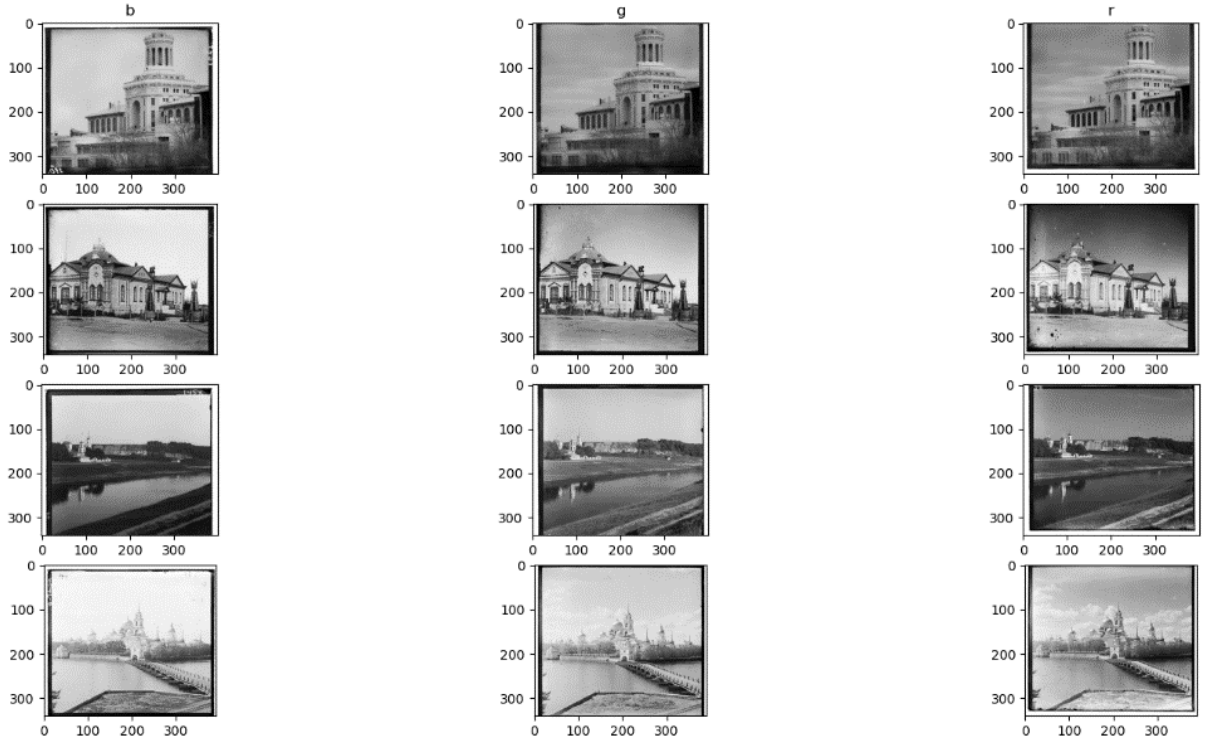
d) کد مربوط به این بخش در فایل d.py قرار دارد. مانند قسمت c، تمام پیکسل‌هایی که رنگ آن‌ها با خانه‌های خاکستری یکسان نیست، حذف می‌کنیم. مطابق شکل ۵ خانه‌های A و B کاملا هم رنگ هستند.



شکل ۵

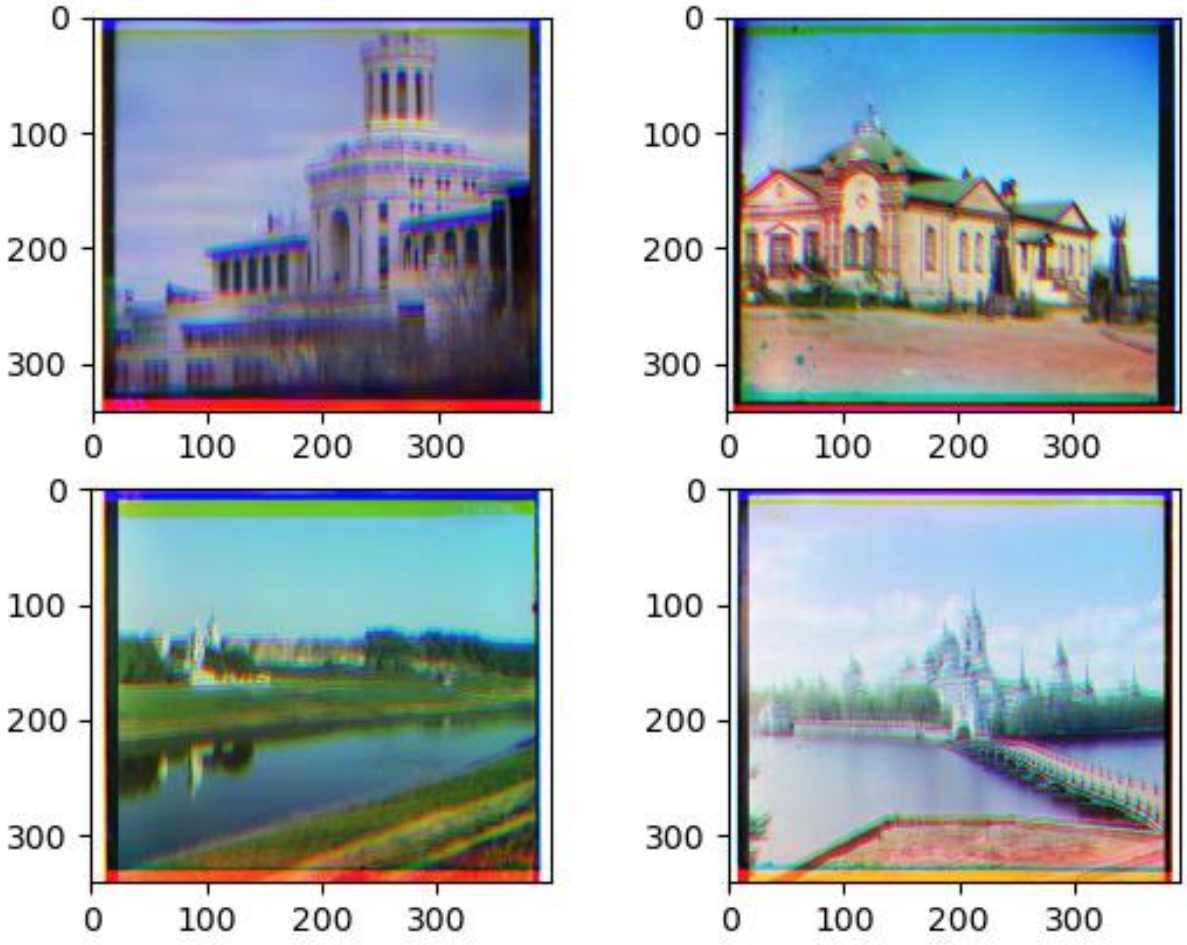
۲)

a) کد مربوط به این بخش در فایل a.py قرار دارد. تابع مورد نظر تصویر ورودی را به سه بخش عمودی مساوی تقسیم و برمی‌گرداند. رنگ کانال‌ها با توجه به خروجی بخش b، با آزمون و خطا پیدا شده‌است. چهار مورد مورد از خروجی تابع در شکل زیر ارائه شده است.



شکل ۶

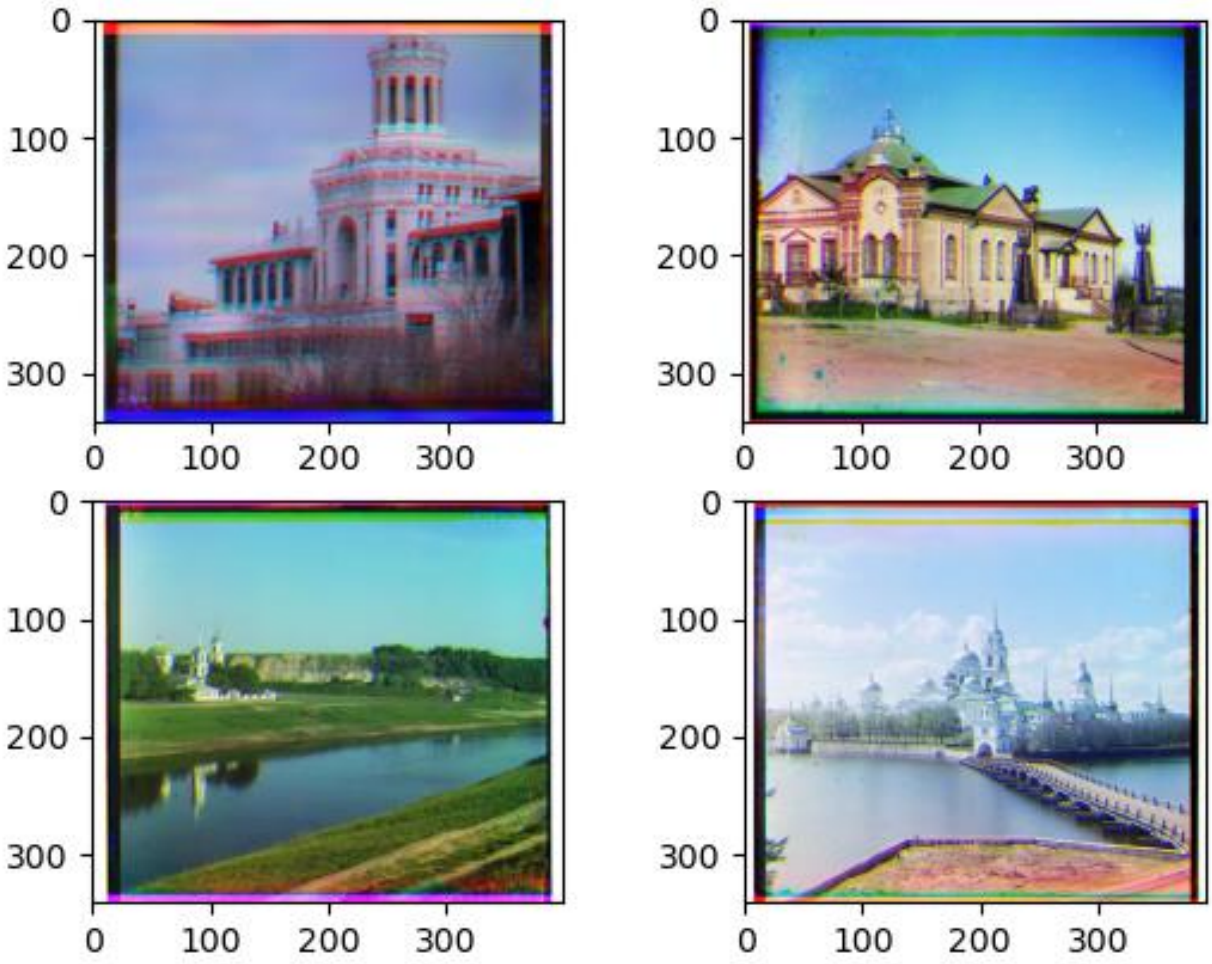
b) کد مربوط به این بخش در فایل b.py قرار دارد. در شکل۷، کانال‌های نشان داده شده در شکل۶، تبدیل به یک تصویر رنگی شده‌اند. مطابق انتظار کانال‌ها کاملا روی هم منطبق نشده‌اند.



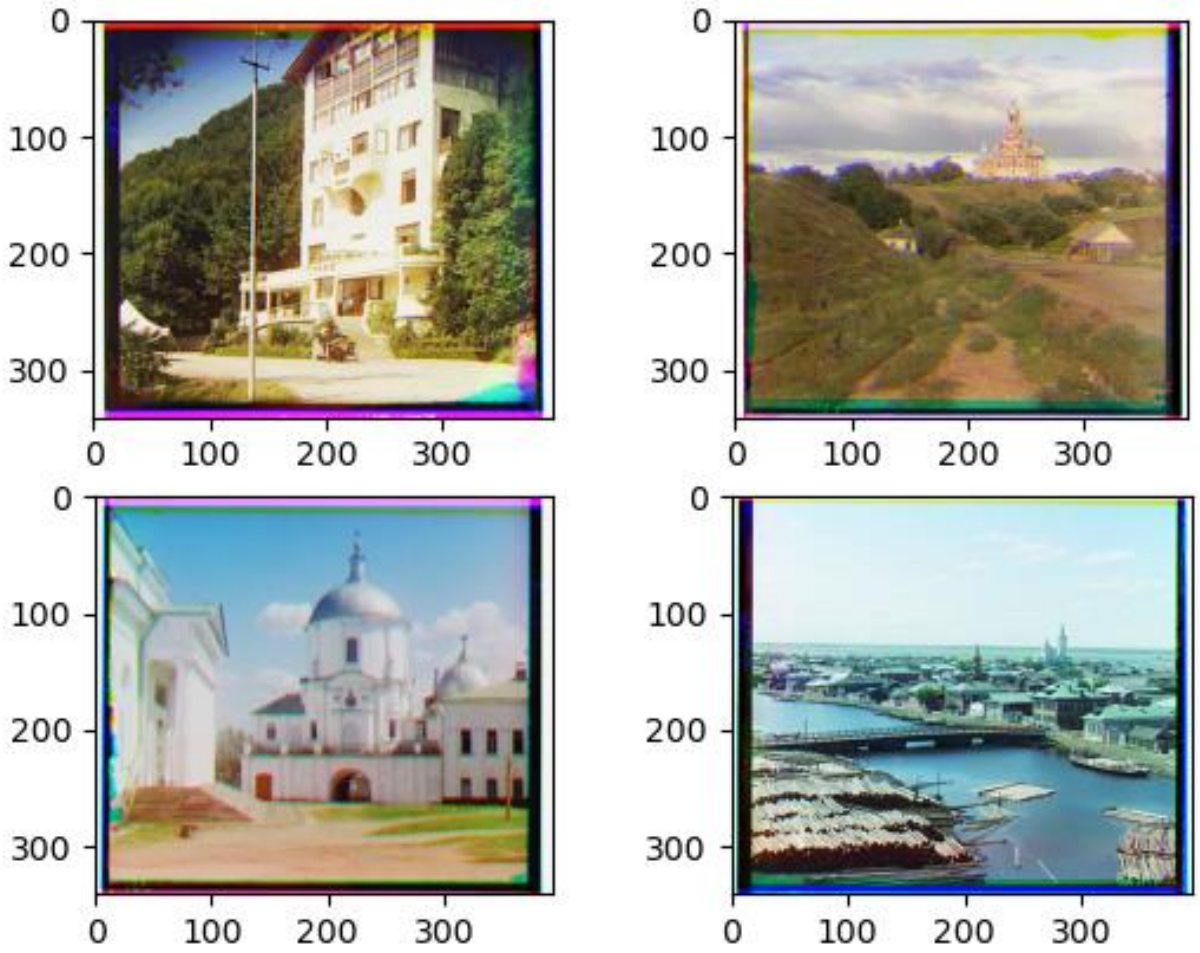
شکل ۷

c) کد مربوط به این بخش در فایل c.py قرار دارد. از مجموع ضرایب هم‌بستگی پیرسون[1] به عنوان معیار شباهت دو تصویر استفاده می‌کنیم. بدین منظور کانال g را ثابت فرض می‌کنیم و دو کانال دیگر را روی آن منطبق می‌کنیم. خروجی در شکل ۸ نشان داده شده است. تنها در تصویر ۱ اندکی عدم انطباق کانال‌ها مشاهده می‌شود و سایر تصاویر به خوبی منطبق شده‌اند. برای نشان دادن صحت عملکرد الگوریتم، در شکل۹، چهار تصویر دیگر نمایش داده شده است.

[1] Pearson's Product-Moment Correlation

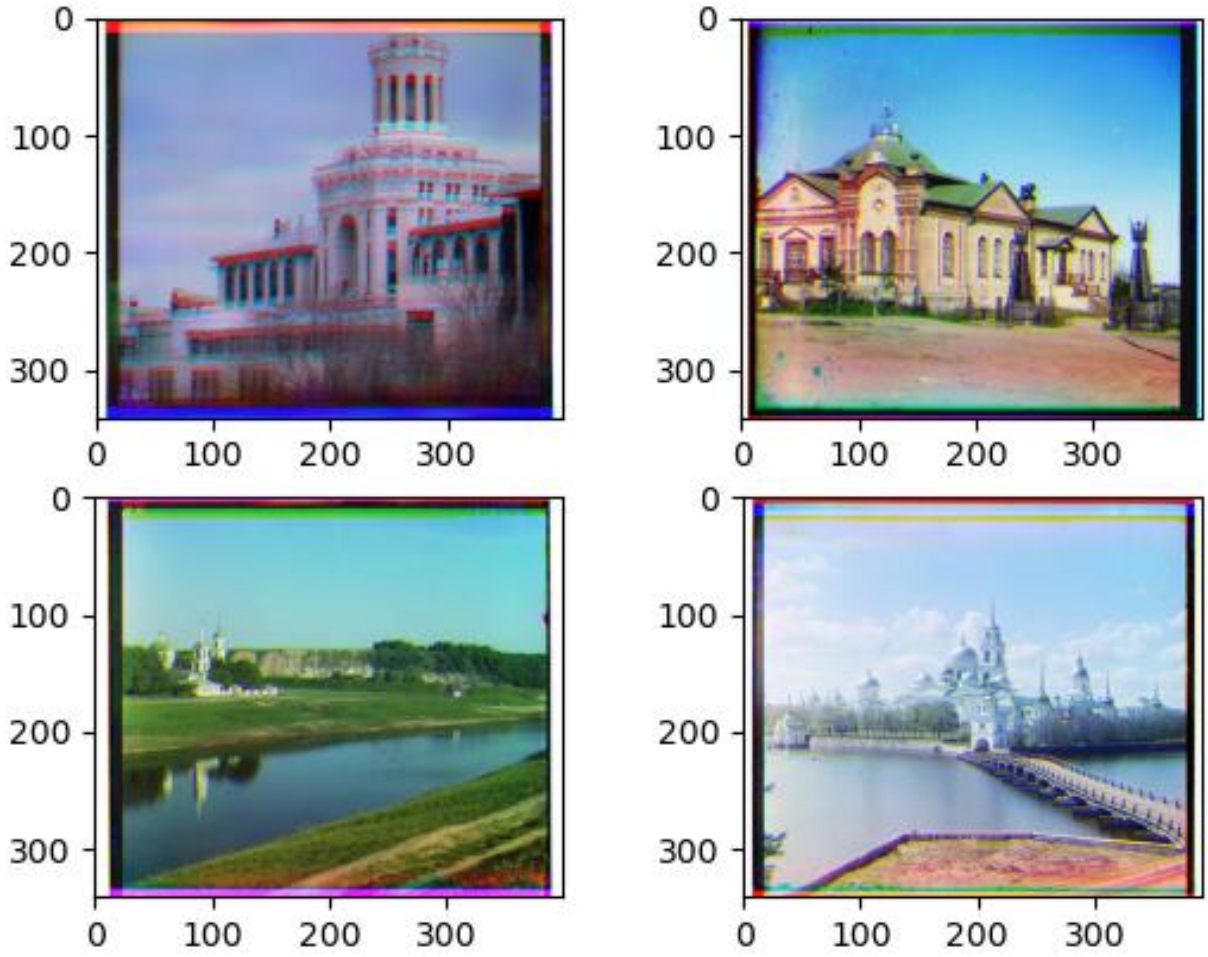

0
100
200
300
0
100
200
300
0
100
200
300
0
100
200
300
0
100
200
300
0
100
200
300
0
100
200
300
0
100
200
300

شکل ۸



شکل ۹

d) کد مربوط به این قسمت در فایل d.py قرار دارد. با اعمال تغییراتی روی توابع قسمت قبل، الگوریتم مورد نظر را پیاده سازی می‌کنیم. اگر فرض کنیم حلقه دو بعدی جستوجو قسمت اصلی برنامه بوده است و n بازه جستوجو باشد؛ در این صورت زمان اجرای برنامه قسمت c برابر با $O(n^2)_{original\_image}$ بوده و زمان اجرای برنامه قسمت d برابر با $O(0.25n^2)_{original\_image}$ + $O(0.25n^2)_{reduced\_image}$ می‌باشد. بنابراین انتظار داریم زمان اجرای برنامه حداقل ۲ برابر افزایش یابد. زمان اجرای این قسمت حدود ۱۳ ثانیه می‌باشد که در مقایسه با زمان اجرای قسمت قبل (۴۳ ثانیه)، حدود ۳ برابر بهبود یافته‌است. خروجی در شکل۱۰ ارائه شده‌است.

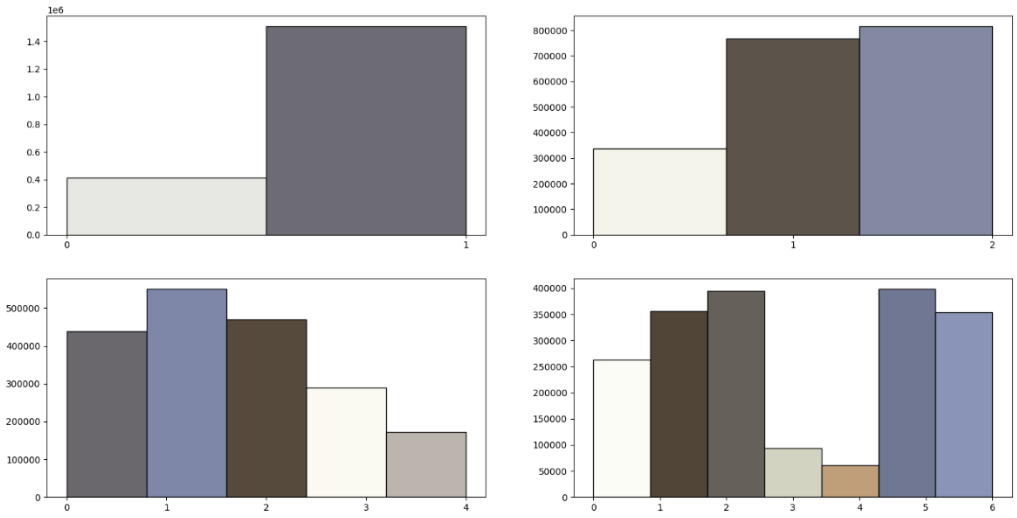


شکل ۱۰

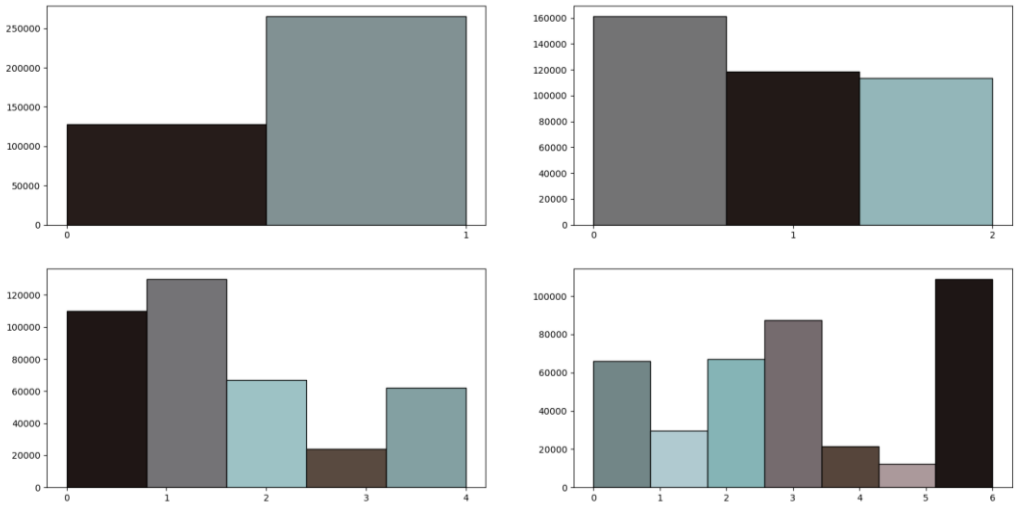
۳)

a) کد مربوط به این بخش در فایل a.py قراردارد. کندترین بخش الگوریتم kmeans پیاده‌سازی شده، محاسبه فاصله هر نقطه از مراکز کلاسترها است. برای افزایش سرعت این بخش، از تابع euclidean_distances کتابخانه scikit-learn استفاده می‌کنیم. این تابع ، دو ماتریس را دریافت کرده و فاصله بین همه نقاط آن‌ها را دوبه‌دو محاسبه می‌کند. این تابع از عملیات ماتریسی استفاده می‌کند و به همین دلیل سرعت آن چند ده برابر نوشتن حلقه‌های دو بعدی است.

با توجه به شکل ۱۱ که خروجی الگوریتم برای تصویر لباس است، رنگ واقعی آن آبی و مشکی می‌باشد. در شکل ۱۲، خروجی الگوریتم برای تصویر nike_outfit مشخص شده است. با توجه به این شکل، رنگ واقعی این لباس خاکستری و سبز مایل به آبی است.(رنگ مشکی مربوط به پس زمینه است)

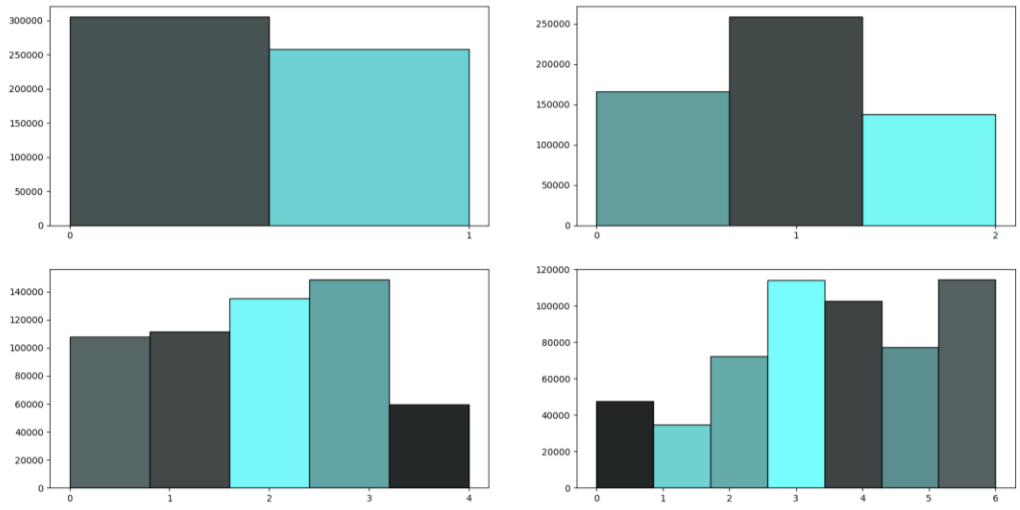


شکل ۱۱

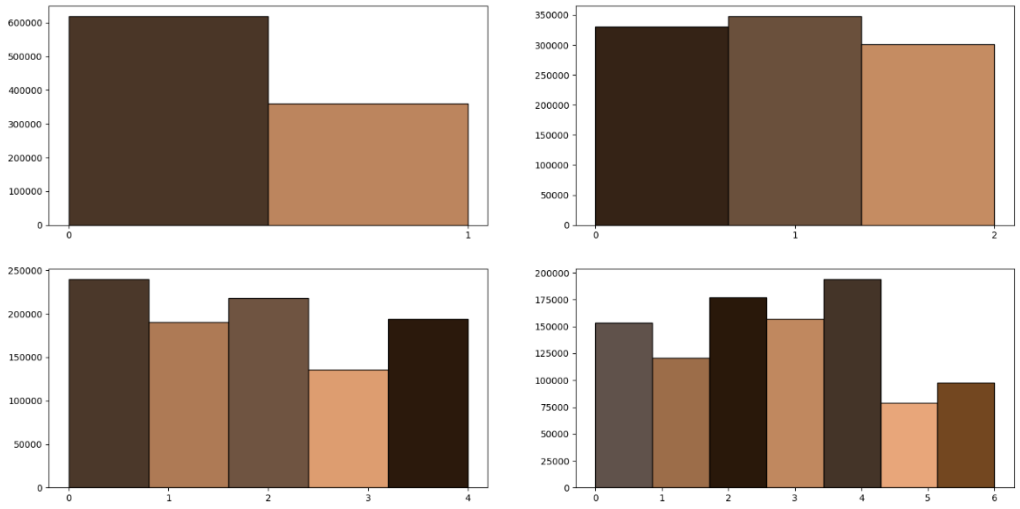


شکل ۱۲

شکل ۱۳ نشان می‌دهد که در تصویر strawberries.png، همه رنگ‌های پرتکرار از طیف سبز و آبی هستند و هیچ رنگ قرمزی در این تصویر وجود ندارد. در نهایت با توجه به شکل ۱۴، هیچ رنگ آبی–سبزی(bulish) در تصویر candy_box وجود ندارد.

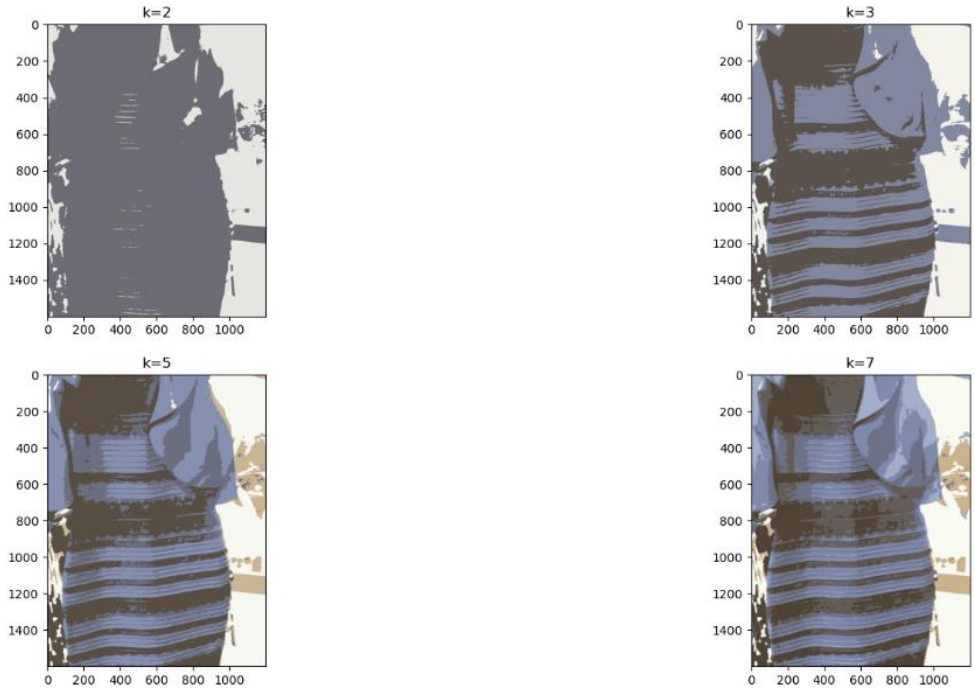


شکل ۱۳

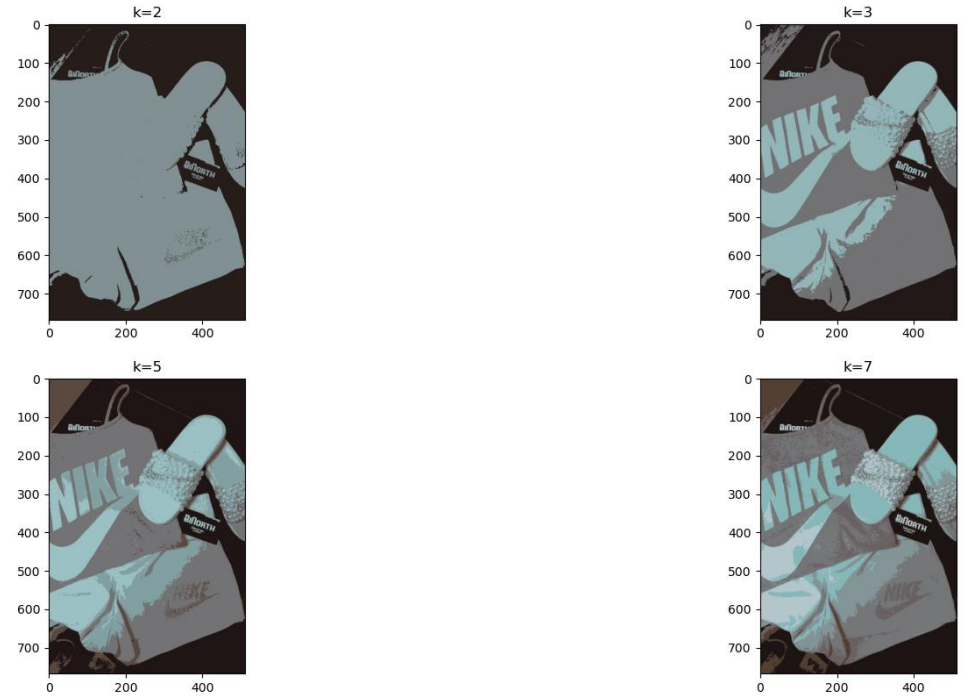


شکل ۱۴

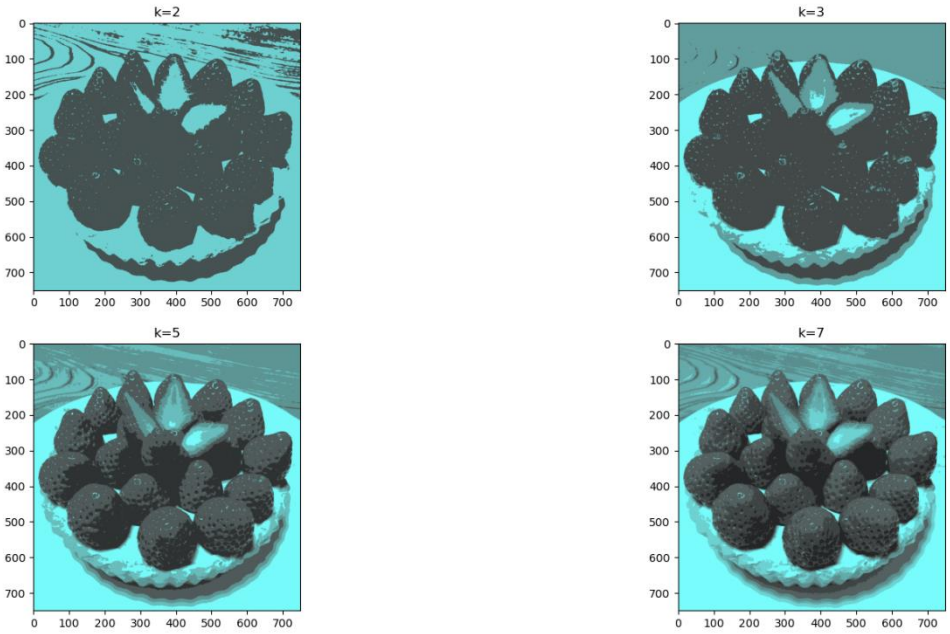
b) کد مربوط به این بخش در فایل b.py قرار دارد. خروجی همه تصاویر به ترتیب در شکل‌های ۱۵ تا ۱۸ ارائه شده‌است. به نظر می‌رسد segment کردن تصاویر تنها در k=2 یا k=3 به دیدن رنگ واقعی تصاویر کمک می‌کند و در k های بالاتر، خطای دید مجددا باز می‌گردد.



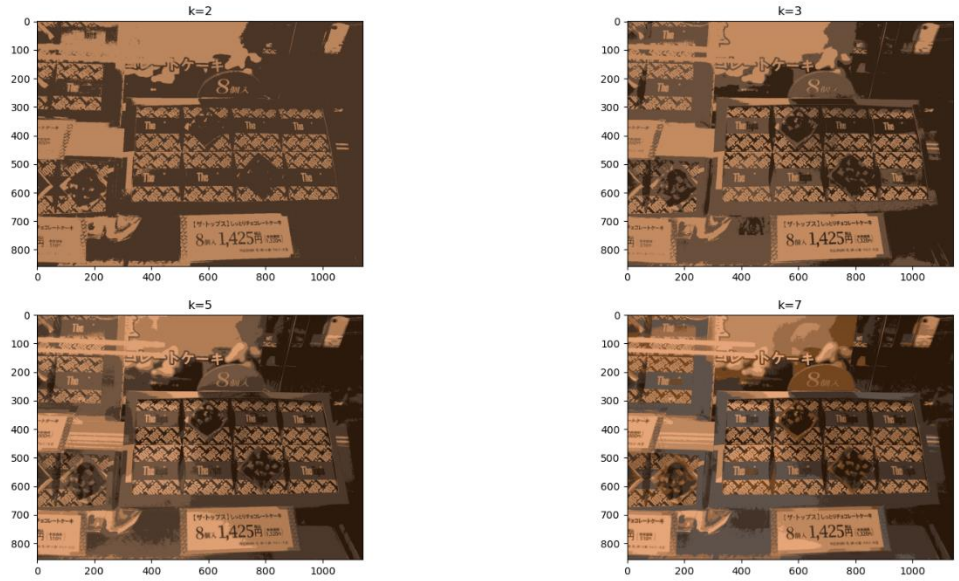
شکل ۱۵



شکل ۱۶



شکل ۱۷



شکل ۱۸

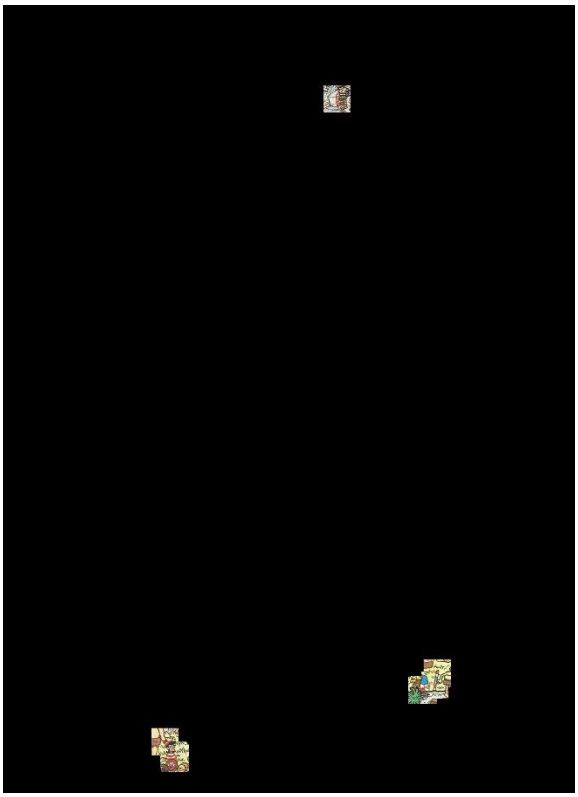
۴)

a) کد مربوط به این قسمت در فایل a.py قرار دارد. لباس wally در تصاویر مختلف دارای رنگ‌های اندکی متفاوت است. به همین دلیل از بازه‌ای از رنگ‌های قرمز و سفید برای مشخص کردن آن استفاده می‌کنیم. برای این کار از الگوریتم kmeans سوال قبل استفاده می‌کنیم. به

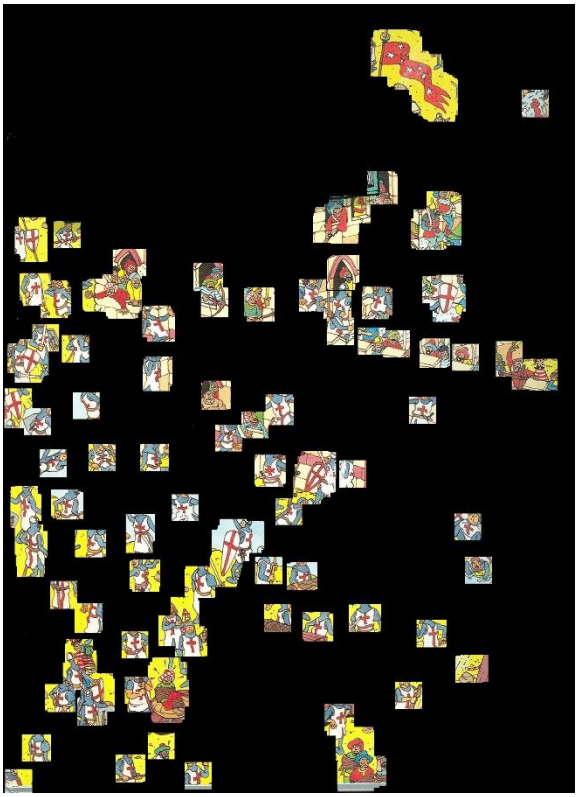
این صورت که پرتکرارترین رنگ سفید و قرمز به ازای k=7 را از یکی از تصاویر wally استخراج می‌کنیم. سپس یک بازه +-۱۰ برای پوشش تغییرات رنگ درنظر می‌گیریم. و از این پس از همان بازه‌ها برای همه تصاویر استفاده می‌کنیم. محدوده‌های انتخاب شده به این صورت است:

```
red1 = (225,70,70)
red2 = (245,90,90)
#white1 =  (254,254,254)
#white2 = (255,255,255)
```

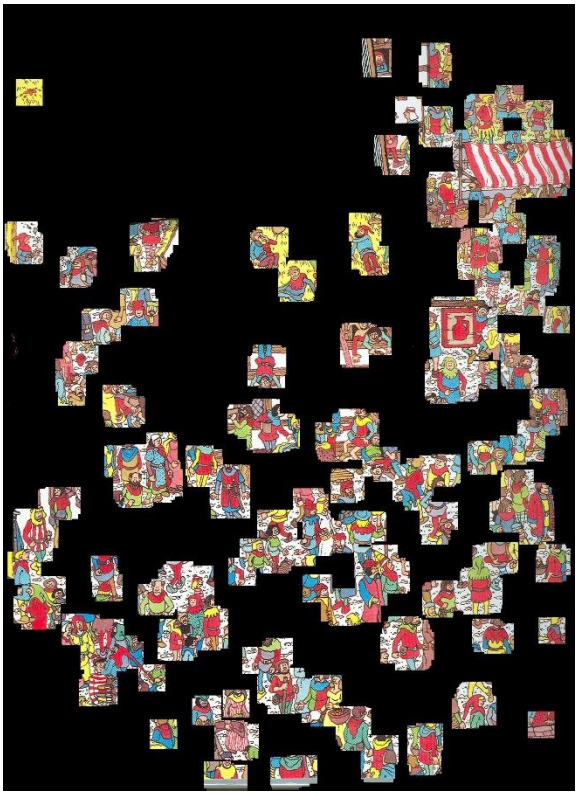
b) کد مربوط به این قسمت در فایل b.py قرار دارد. با توجه به توضیحات قسمت قبل و در نظر گرفتن یک بلاک مربعی به ضلع ۸۰ حول پیکسل‌های مجاز، خروجی ها در شکل‌های ۱۹ تا ۲۱ نشان داده شده است.(این تصاویر همچنین در پوشه مربوط به این سوال ذخیره شده‌اند) لازم به ذکر است برای انتخاب پیکسل‌های مجاز،به دو دلیل تنها از بازه قرمز رنگ استفاده شده‌است؛ اول، تعداد پیکسل‌های سفید در عکس‌ها بسیار زیاد است و حجم محاسبات را افزایش می‌دهد. دوم، با درنظر گرفتن بلاک به ضلع ۸۰ حول پیکسل‌های قرمز، عملا پیکسل‌های سفید موجود در لباس wally نیز در خروجی ظاهر خواهند شد.



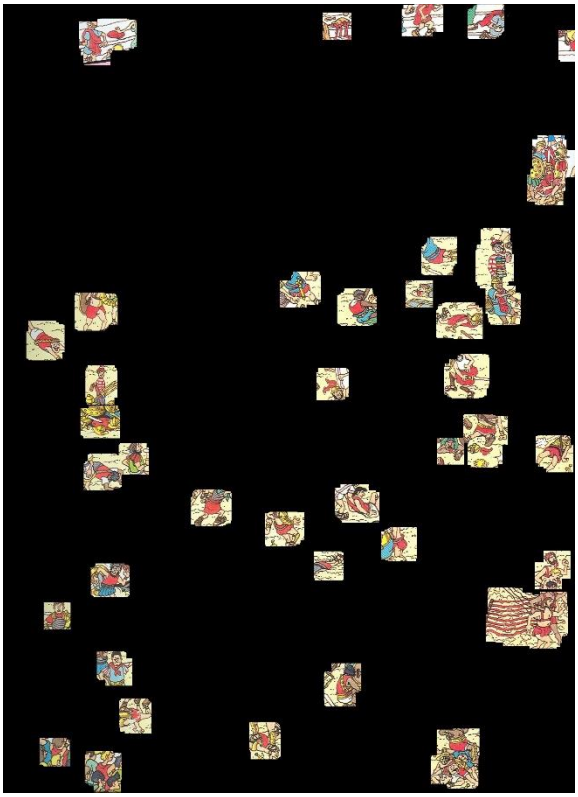
شکل ۱۹



شکل ۲۰



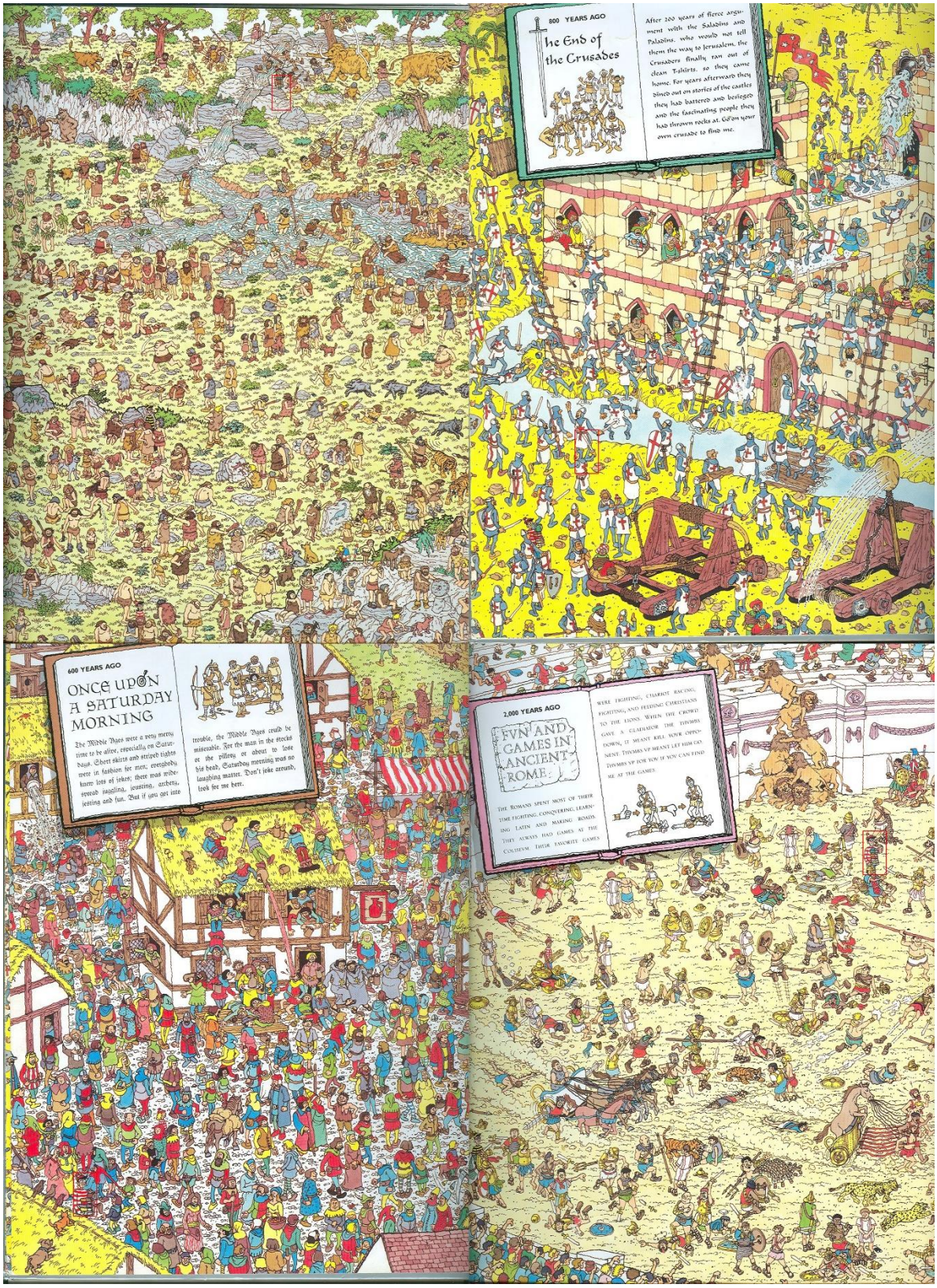
شکل ۲۱



شکل ۲۲

c) کد مربوط به این قسمت در فایل c.py قرار دارد. در خط ۶ و ۹ به ترتیب آدرس تصویر کاهش یافته(خروجی قسمت b) و آدرس تصویر اصلی متناظر با آن وارد می‌شود. از آنجا که حجم محاسبات، حتی در تصویر کاهش یافته زیاد است. تنها از یک قالب (wally_2.jpg) و دو اندازه (۱۰۰ درصد و ۸۰ درصد) استفاده می‌کنیم.به هرحال برنامه به گونه‌ای نوشته شده است که هر تعداد قالب و اندازه را اجرا کند. برای افزایش بیشتر سرعت، در خط ۵ متغیری به نام stride تعیین شده که گام عبور از پیکسل‌های افقی را مشخص می‌کند. به جز تصویر ۱ که در آن wally ابعاد بسیار کوچکی دارد، در سایر تصاویر می‌توان آن را ۵ تعیین کرد. خروجی ها در بخش d ارائه خواهد شد.

d) شکل ۲۳ محل wally را در همه تصاویر نشان می‌دهد.(کادر قرمز رنگ) این شکل‌ها به صورت مجزا در پوشه مربوط به این سوال ذخیره شده‌اند.(فایل های c1.jpg تا c4.jpg) همانطور که مشاهده می‌شود، به جز تصویر ۲، در سایر تصاویر، الگوریتم به درستی عمل کرده است. برای بهبود نتیجه تصویر ۲ می‌بایست قالب‌ها و ابعاد بیشتری را به الگوریتم اضافه کرد.


800 YEARS AGO
The End of the Crusades
After 200 years of fierce argument with the Saladins and Paladins, who would not tell them the way to Jerusalem, the Crusaders finally ran out of clean T-shirts, so they came home. For years afterward they pined out on stories of the castles they had battered and besieged, and the fascinating people they had thrown rocks at. Go'on your own crusade to find me.
600 YEARS AGO
ONCE UPON A SATURDAY MORNING
The Middle Ages were a very merry time to be alive, especially on Saturdays. Short skirts and striped tights were in fashion for men, everybody knew lots of jokes, there was widespread juggling, jousting, archery, feasting and fun. But if you got into trouble, the Middle Ages could be miserable. For the man in the stocks or the pillory, or about to lose his head, Saturday morning was no laughing matter. Don't joke around, look for me here.
2,000 YEARS AGO
FUN AND GAMES IN ANCIENT ROME
The Romans spent most of their time fighting, conquering, learning Latin and making roads. They always had games at the Coliseum. Their favorite games were fighting, chariot racing, fighting, and feeding Christians to the lions. When the crowd gave a gladiator the thumbs down, it meant kill, your opponent. Thumbs up meant let him go. Thumbs up for now if you can find me at the games.

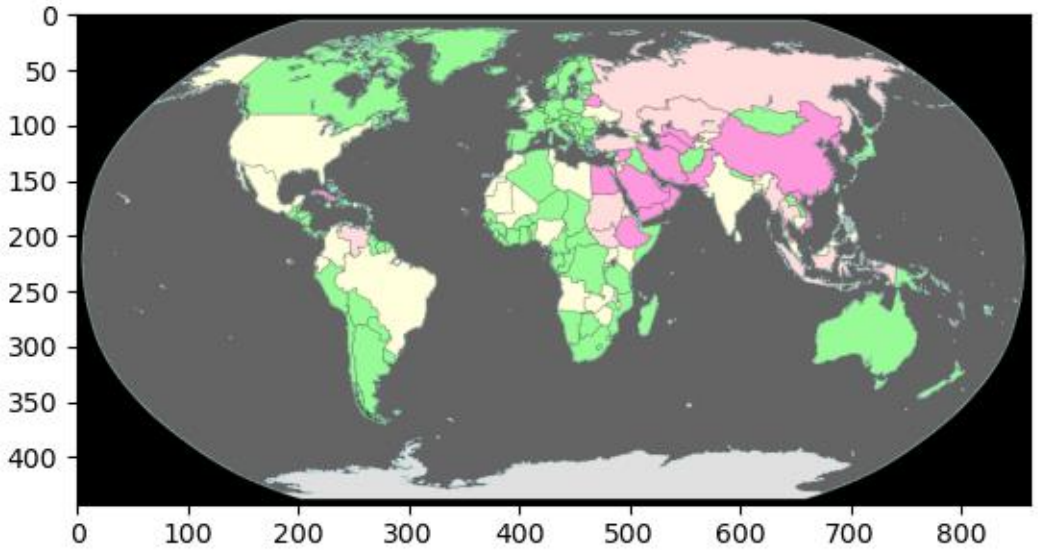
شکل ۲۳

۵)

a) کد مربوط به این قسمت در فایل a.py قرار دارد. این تصویر دارای فرمت RGBA می‌باشد. در تمامی سوال‌های مربوط به این تصویر، ابتدا مساحت ناحیه خارج از نقشه که مقدار کانال A آن صفر است، استخراج می‌کنیم. برای محاسبه مساحت کل زمین، این مساحت را از مساحت کل تصویر کم می‌کنیم.

با توجه به رنگ مربوط به آب‌ها، ناحیه آب‌های کره زمین را انتخاب و درصد آن را محاسبه می‌کنیم. جهت اطمینان از درستی ناحیه انتخاب شده، آن ناحیه با رنگ خاکستری در شکل ۲۴ نمایش داده‌شده است.

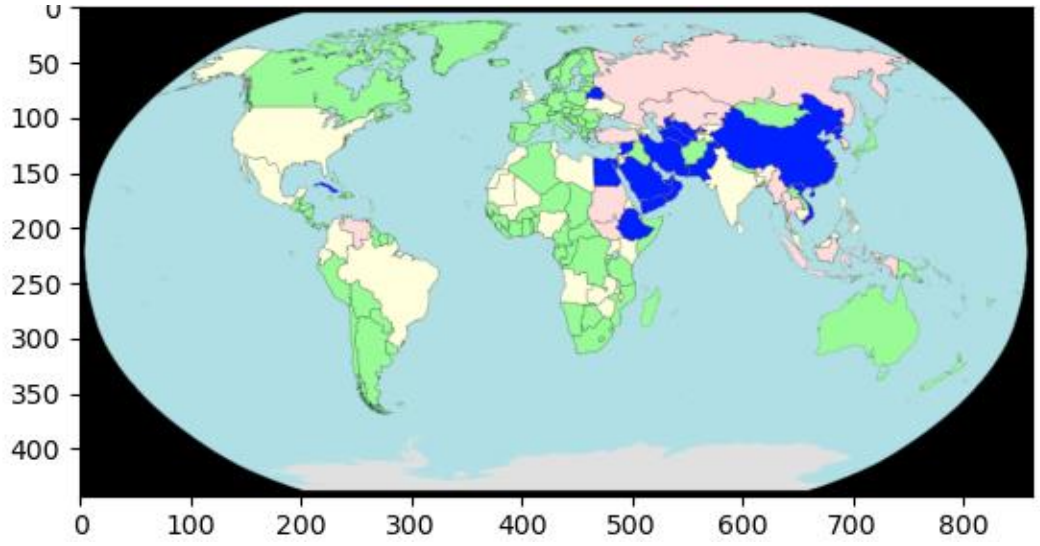
```
percentage of water on earth: 66.20753637530345
```



شکل ۲۴

b) کد مربوط به این قسمت در فایل b.py قرار دارد. نواحی کشورهای موردنظر در شکل ۲۵ با رنگ آبی نشان داده شده و درصد این کشورها(نسبت به کل خشکی‌ها) به شرح زیر است. لازم به ذکر است برای پوشش دادن تغییرات رنگ به خصوص در مرزها، یک بازه از رنگ برای انتخاب این نواحی انتخاب شده است.

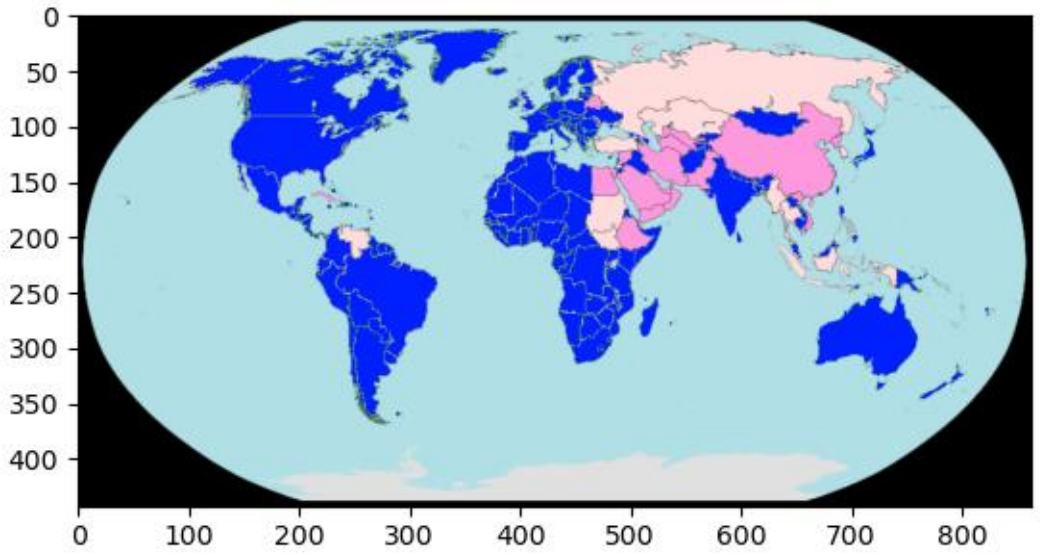
```
percentage of countries with pervasive censorship:  9.201724122152774
```



شکل ۲۵

c) کد مربوط به این قسمت در فایل c.py قرار دارد. نواحی مد نظر در شکل ۲۶ با رنگ آبی نمایش داده شده‌اند. خروجی به این صورت است:

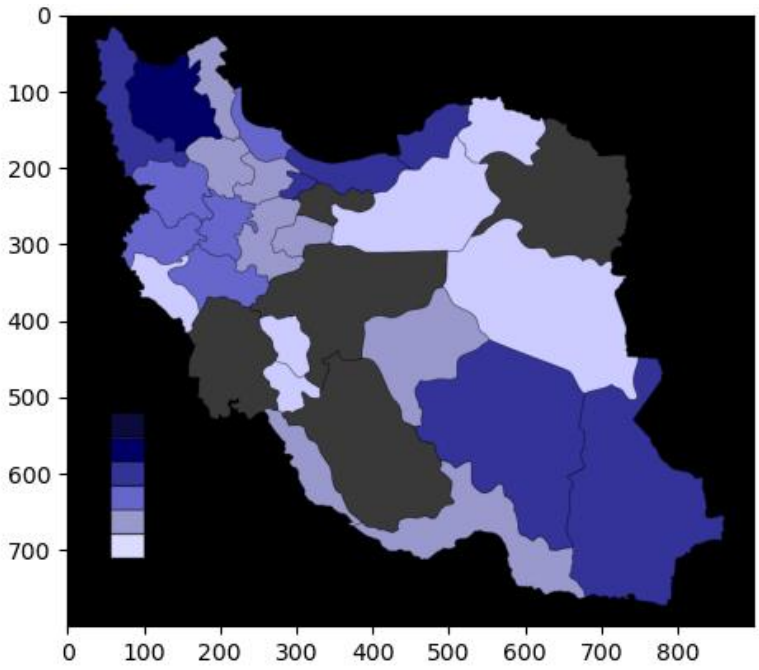
```
percentage of countries with selective or little censorship:
41.23616995964237
```



شکل ۲۶

d) کد مربوط به این بخش در فایل d.py قرار دارد. روند حل سوالات مربوط به این شکل نیز مانند شکل قبل است، بنابراین از ذکر جزئیات صرف‌نظر می‌شود.

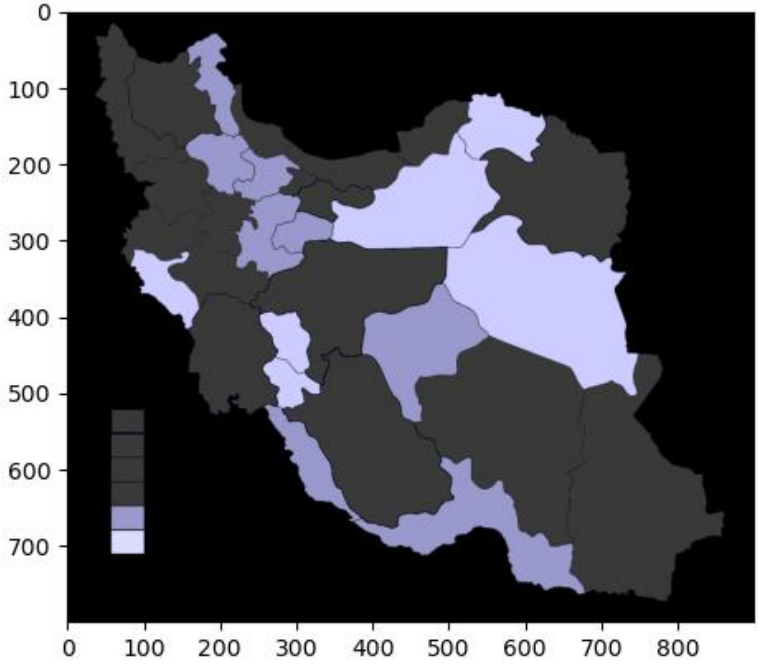
```
percentage of provinces with more than 4 million population:
24.839061707709607
```



شکل ۲۷

e) کد مربوط به این قسمت در فایل e.py قرار دارد.

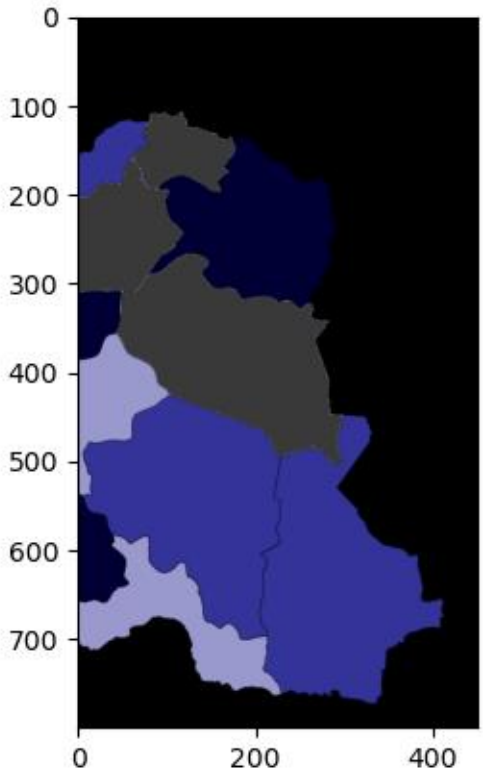
```
percentage of provinces with more than 2 million population:
61.36764478934721
```



شکل ۲۸

f) کد مربوط به این قسمت در فایل f.py قرار دارد. بدین منظور نقشه ایران را به دو قسمت تقسیم کرده و فقط بخش شرقی آن را در نظر می‌گیریم. روند حل مانند قسمت‌های قبل است.

```
percentage of provinces with less than 1 million population in the eastern part:  27.25600889573176
```

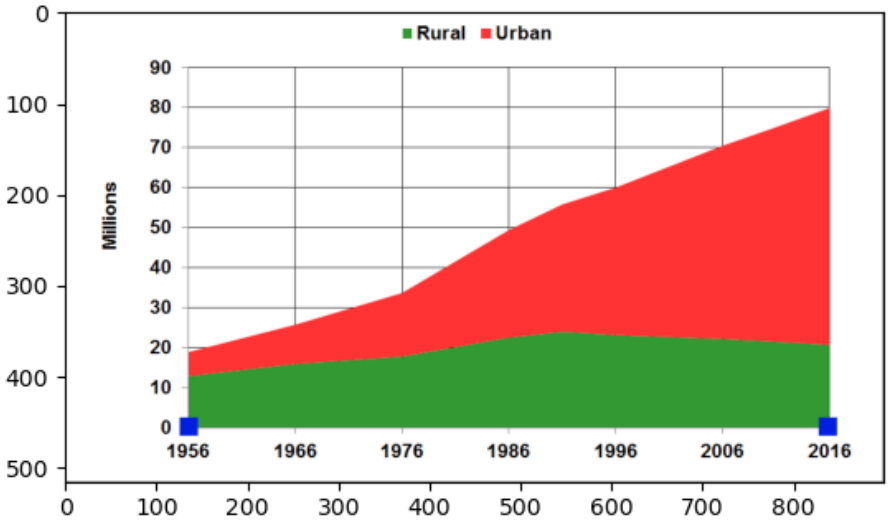


شکل ۲۹

g) کد مربوط به این بخش در فایل g.py قرار دارد. مقایسه جمعیت شمال و جنوب با اطلاعات موجود دشوار است. زیرا جمعیت هر استان به مساحت آن وابسته نیست و باید تعداد استان‌های هر رنگ را حساب کنیم. اما به صورت تقریبی مساحت هر بخش را به صورت مجموع درصد هر رنگ در جمعیت آن در نظر می‌گیریم. با این فرض نه چندان صحیح و با تقسیم نقشه از وسط به دو نیم، خروجی به صورت زیر است:

```
South has more population
```

h) کد مربوط به این بخش در فایل h.py قرار دارد. قبل از شروع کل سوالات مربوط به این عکس، انجام دو عمل ضروری است. اول، خطوط گرید موجود در نمودار را با روشی شبیه به Image Thresholding پاک می‌کنیم. دوم، محدوده شروع و پایان نمودار را با توجه به موقعیت نقاط و رنگ پیکسل آن‌ها تعیین می‌کنیم.(به عنوان مثال نقطه شروع، نقطه سبز رنگی است که بیشترین y و کمترین x را نسبت به مرکز مختصات دارد) محدوده کل نمودار با دو نقطه آبی در شکل ۳۰ نشان داده شده است.



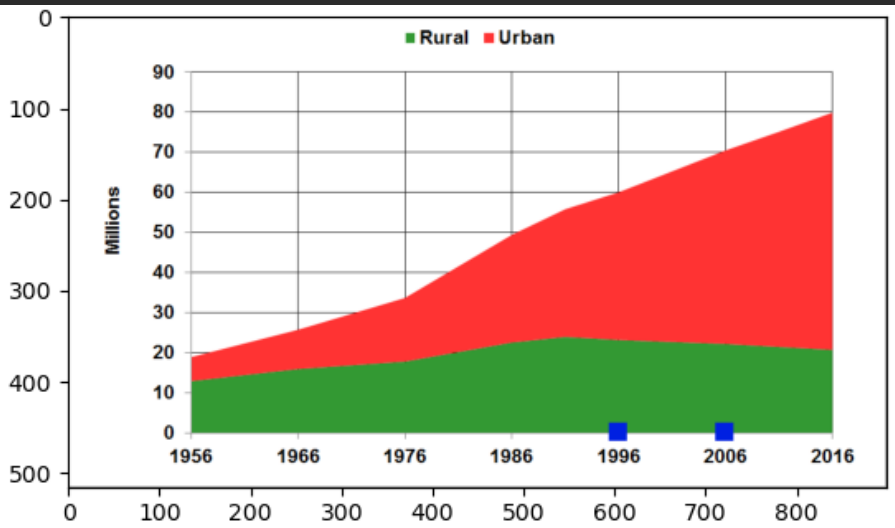
شکل ۳۰

حال به بررسی خواسته سوال می‌پردازیم. برای بدست آوردن نسبت خواسته شده، تابعی با عنوان find_y_values نوشته شده که با گرفتن نقطه x از نمودار، ارتفاع منحنی سبز و قرمز را برمی‌گرداند. با اجرای یک حلقه روی محدوده بین دو نقطه آبی در شکل ۳۰، مجموع جمعیت شهری و کل را در بازه بدست آورده و بر هم تقسیم می‌کنیم. خروجی به این صورت است:

```
ratio:  0.590291786135942
```

i) کد مربوط به این قسمت در فایل i.py قرار دارد. محدوده مدنظر را یک بار جهت اطمینان در شکل ۳۱ نمایش می‌دهیم. (این محدوده با توجه به اطلاعات کل بازه و عمل تقسیم و جمع بدست آمده است) با روشی شبیه به بخش h، خروجی به این صورت است:

```
ratio between 1996 to 2006:  0.34659225837093327
```

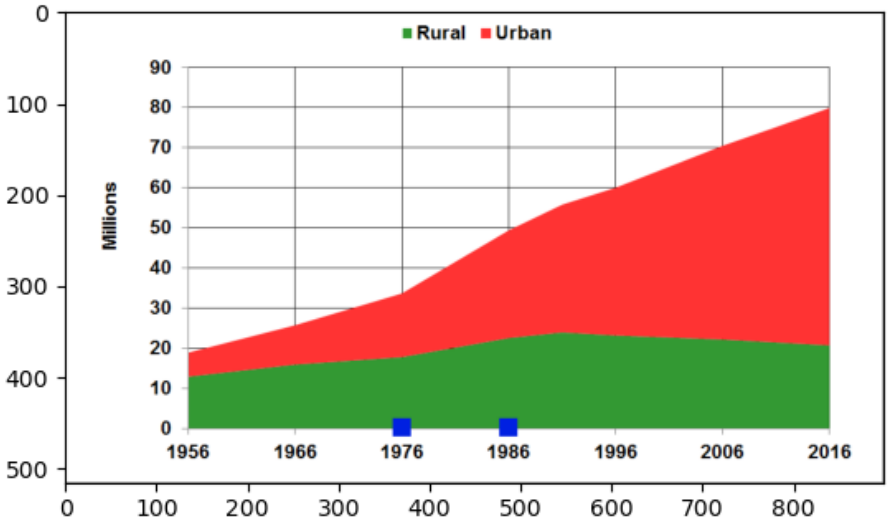


شکل ۳۱

j) کد مربوط به این قسمت در فایل j.py قرار دارد. با تقسیم بازه‌ای که در شکل ۳۲ نشان داده شده است، به ده قسمت مساوی، جمعیت شهرنشین را در هر سال از دهه موردنظر و همچنین مجموع جمعیت این ده سال را محاسبه می‌کنیم.

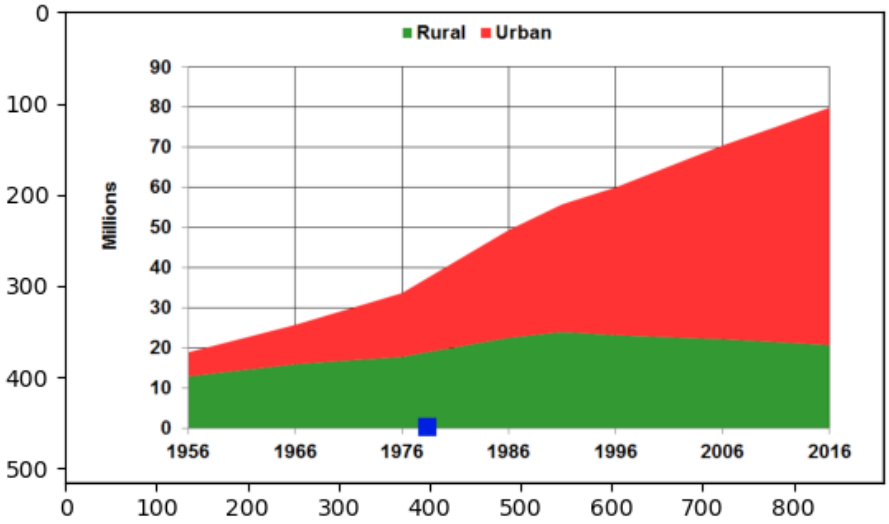
```
population in  1976 :  15.949367088607591
```

```
population in  1977 :  17.08860759493671
population in  1978 :  18.0
population in  1979 :  19.139240506329113
population in  1980 :  20.050632911392405
population in  1981 :  21.189873417721518
population in  1982 :  22.101265822784807
population in  1983 :  23.0126582278481
population in  1984 :  24.15189873417722
population in  1985 :  25.0632911392405
total urban population in 1976:1986 : 205.74683544303798
```



شکل ۳۲

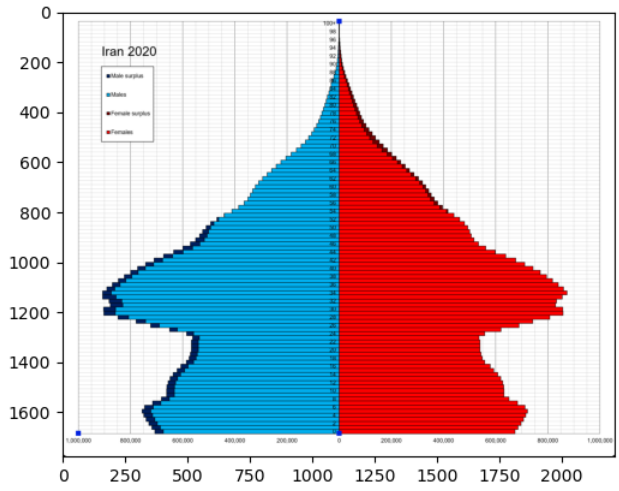
k) کد مربوط به این قسمت در فایل k.py قرار دارد. با در نظر گرفتن شرط توقف abs(y_urban - y_rural) < 0.00001 برای توقف حلقه، محل تقریبی برابر شدن جمعیت شهری و روستایی در شکل ۳۳ نمایش داده شده است.



شکل ۳۳

l) کد مربوط به این بخش در فایل l.py قرار دارد. سه نقطه مهم برای حل سوالات مربوط به این شکل، در شکل ۳۴ نمایش داده شده است. این نقاط با توجه به رنگ پیکسل‌ها و محل نسبی آن‌ها بدست آمده است. برای بدست آوردن جمعیت‌ها، یک تابع برای استخراج جمعیت در هر نقطه از نمودار نوشته و با فواصل مساوی برای هر سن اجرا می‌شود. در نهایت مجموع جمعیت‌ها به شرح زیر است. بنابراین جمعیت مردان اندکی از زنان بیشتر است.

```
male population: 41629913.71045064
female population: 41524448.70565677
```



شکل ۳۴

m) کد مربوط به این بخش در فایل m.py قرار دارد. جمعیت کل و جمعیت افراد بالای ۵۰ را محاسبه و برهم تقسیم می‌کنیم.

```
 percentage of over 50 17.900380491179536
```

n) کد مربوط به این بخش در فایل n.py قرار دارد. این سوال نکته خاصی ندارد و مشابه بخش l حل می‌شود.

```
male population: 13397890.699904123
female population: 13378715.244487055
```

o) کد مربوط به این بخش در فایل o.py قرار دارد. جمعیت هر سال به تفکیک، و مجموع جمعیت کل دهه محاسبه شده است.

```
number of 40 years old mens: 744966.4429530202
number of 41 years old mens: 713326.941514861
number of 42 years old mens: 678811.121764142
number of 43 years old mens: 640460.2109300096
number of 44 years old mens: 600191.7545541706
number of 45 years old mens: 563758.3892617449
number of 46 years old mens: 535953.978906999
number of 47 years old mens: 518696.06903163943
number of 48 years old mens: 508149.5685522531
number of 49 years old mens: 502396.9319271333
total male population between 40-50: 6006711.409395972
```

به علت زیاد بودن تعداد تصاویر خروجی، از نمایش آن‌ها در این گزارش صرف‌نظر می‌کنیم. کد مربوط به هر قسمت و ویدیو مربوط به آن در پوشه مربوط به این سوال قرار دارد. در انتها، در فایل f.py، فایل‌های ویدیویی باهم ترکیب و در قالب یک فایل ارائه شده است. برای ایجاد و ویرایش ویدیوها از کتابخانه moviepy استفاده شده است. لازم به ذکر است پس از اجرای هر کد، فایل ویدیویی مربوط به آن در فایلی با عنوان __temp__ ذخیره می‌شود.

- a.mp4
- b.mp4
- c.mp4
- d.mp4
- e.mp4
- f.mp4

۷)

---

a) این پدیده که به پس‌دید منفی[۲] معروف است. زمانی رخ می‌دهد که سلول‌های گیرنده چشم، به ویژه سلول‌های مخروطی، نسبت به یک ورودی بیش از حد انطباق یافته و حساسیت خود را از دست می‌دهند. به طور معمول در دنیای واقعی این اتفاق برای ما نمی‌افتد. زیرا حرکات هرچند کوچک چشم، مانع از انطباق بیش از حد سلول‌ها می‌شود.

b) از بین تصاویر A و B، تصویر B که ابعاد بزرگتری دارد کیفیت بیشتری دارد. همچنین تصویر D به علت وجود اطلاعات رنگی بیشتر، از تصویر C با کیفیت تر است. (سایر ویژگی‌های عکس‌ها یکسان فرض شده‌است)

c) اگر منظور صرفا ساخت یک تصویر با سه کانال رنگی است، بله با تکرار مقادیر طیف خاکستری برای هر سه کانال به یک تصویر RGB خواهیم رسید که البته همچنان خاکستری خواهد بود. اما اگر هدف بدست آوردن تصویر رنگی باشد، این‌کار با روش‌های عادی پردازش تصویر ممکن نیست؛ زیرا مقادیر هر کانال متفاوت است و اطلاعات آن وجود ندارد. در صورتی که تصویری مشابه(نه یکسان) به همراه کانال‌های RGB آن در دسترس باشد، می‌توان از آن برای نگاشت طیف خاکستری به کانال‌های RGB استفاده کرد. اگرچه نتیجه به‌ندرت قابل قبول است.

d) برخی از فضاهای رنگی، در عمق رنگ‌های آن‌ها متفاوت هستند؛ بنابراین برخی از آن‌ها دارای کیفیت بالاتر هستند. دلیل دیگر وجود فضاهای رنگ مختلف، محدودیت دستگاه‌های خروجی نمایش‌دهنده آن‌ها است(چاپگر، نمایشگر و ... دارای تکنولوژی متفاوتی هستند و درنتیجه، فضای رنگی متفاوتی نیاز دارند). برخی فضاهای رنگی نیز برای توصیف رنگ‌ها توسط انسان مناسب‌تر هستند؛ به عنوان مثال فضای رنگ HSV، رنگ‌های مشابه را در کنار هم دسته‌بندی می‌کند.

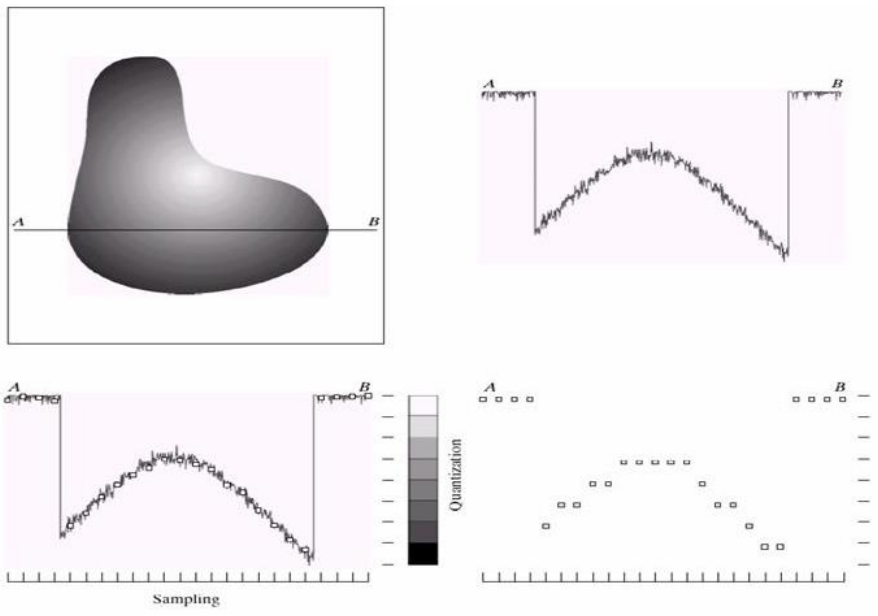
e) دیجیتال‌سازی یک سیگنال، یا به طور خاص یک تصویر، از دو بخش کلی تشکیل شده‌است:

۱) گسسته‌سازی: خواندن اطلاعات با یک فرکانس یا فاصله خاص،که نمونه برداری نامیده می‌شود. در مورد تصاویر، به عنوان مثال با فواصل مشخص و گام‌های عمودی، یک خط افقی از شدت‌ها را بخوانیم. فاصله گام‌ها، تراکم پیکسلی را مشخص می‌کند. (شکل ۳۴–دو تصویر بالا)

۲) کمی‌سازی: نمونه‌ها را به یک مجموعه محدود از اعداد نگاشت می‌کنیم. به عنوان مثال برای رنگ‌های یک تصویر ۲۵۵ عدد گسسته تعریف کنیم. تعداد اعضای مجموعه، عمق رنگ‌ها را مشخص می‌کند. (شکل ۳۴–دو تصویر پایین)

---

[2] Negative afterimages



شکل ۳۵