

بسمه تعالى

پردازش تصویر
تمرین دوم

علی صفرپور دهکردی
۹۹۲۱۰۳۰۱

2	سوال ۱
2	الف
3	ب
6	ج
6	سوال ۲
6	الف
7	ب و ج
9	د
10	ه
12	سوال ۳
12	الف
17	ب
21	ج
28	د

سوال ۱

الف

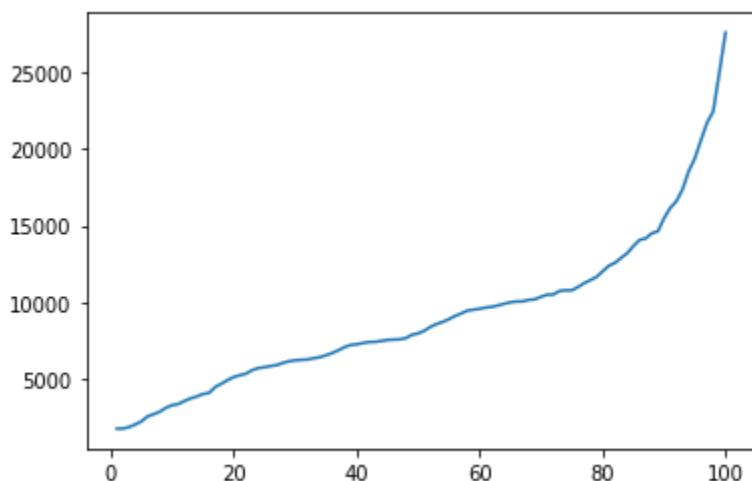
۱- الف)

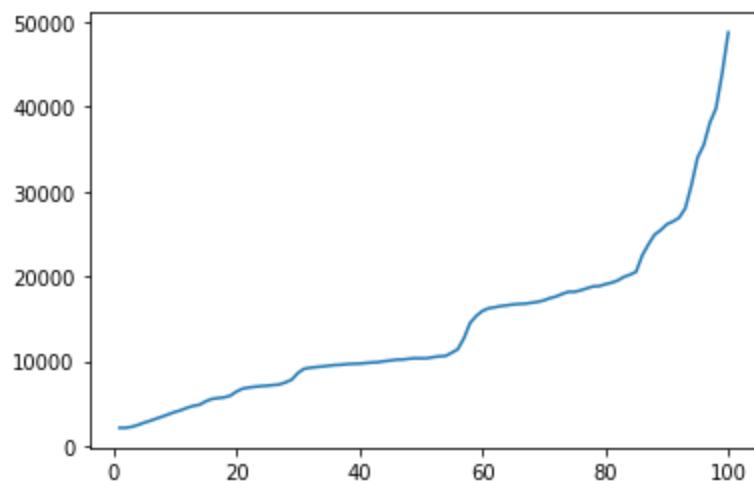
ترنسفورمین استفاده شده DCT است که طبق تعریف راهی بینهایت
از طرفی عکس اعلیٰ بعده است به غیره را محدود نمایند.
کارهای karnen-loeve فوریه (DFT) عکس مخصوص محدود که DCT ندارد.

آنچه آن املازه محدود است دلخواه ارائه می‌کند کافی شناخته نمایند
و بزرگتر از آن locality را در دست یابند سه املازه محدود و کارهای
البته محدود transform در فضای گلی مع اشاره شده است در متابع اینترنت که بعید از
مدتق طراح محترم بوده باشد.

ا- ب) کلی از ازات سه و پنجم ، مشترکه سازی آن است که ابته اصطلاح "عواید" است. این میزان به عبارت "عواید Quality" را نیز می‌نماید. "اصولاً" ثبت تصریر دارای جرئت است که چشم ما یا تشخیص نمایند کم امیت هستند می‌توان این جرئت را که همان تغییر است که در فرمائیس ۴۰ براحتی را حذف نمود و هر چیز کاملاً جم بیشتری متناسب باشد مولفه ها را بیشتری حذف (تصویر) می‌گذارد و می‌توان از بازگرداندن تصریر، جرئت را که در جم کثر شاهد خواهیم شد. ثابت تغییر درجه لذتی است (ناتایی) و همچنین تغییر در (نکره افتی) کدر کمودر در تصریر میان آنها است.

البته تغییر حجم در نمونه‌های مختلف تفاوت دارد که وابسته به تصویر است. در دو تست مختلف داریم:

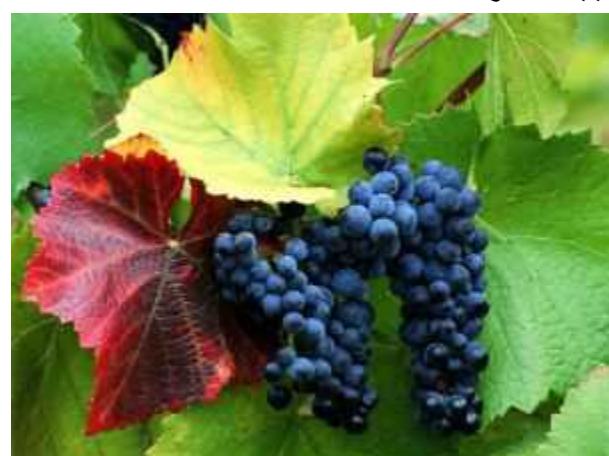




کیفیت ۱۰۰ درصد:



کیفیت ۵۰ درصد:



کیفیت ۱۰ درصد:



کیفیت ۱ درصد:



کدهای مربوطه در نسخه ipny موجود هستند و به علت عدم وجود نکته خاص از آوردن آن پر هیز شد. البته دستور تعیین کیفیت تصویر و سپس محاسبه اندازه آن برای یک عدد بین ۱ تا ۱۰۰ به شرح زیر است:

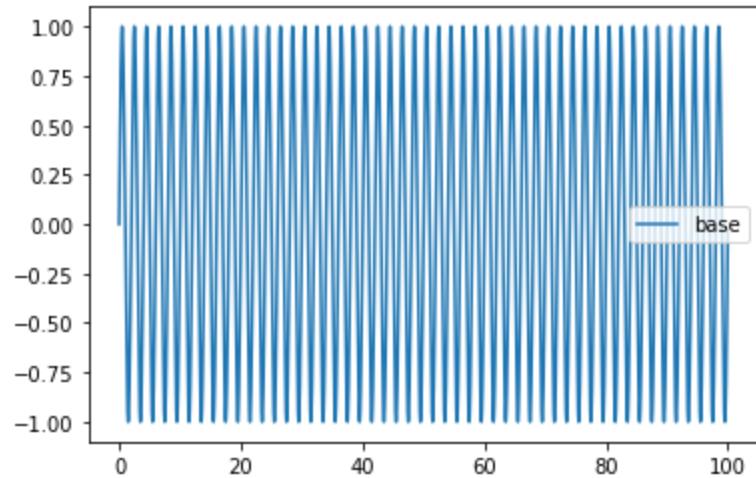
```
# i is a number
image_file.save(str(i)+".jpeg", quality=i)
l.append(os.path.getsize(str(i)+".jpeg"))
```

ا-ج) در تصویر آفرینشی، به موضوع مریع) قابل مشاهده آن رنگ
کلیداً است دارند، حال غرض کنیه بلکه کل تصریر باشد. این اسکه که
کل تصریر بدون درنظر گرفتن جزئیات ^{local}، به است
کلیداً فتن (سایرین شوند) نیست. میان اول گروه جزئیات از دست فتن
مشتث به حالت استاندار ناساب تراست اما محدود نیست اما انت کیفیت
کامرانی خوبی تر خواهد بود. تفاوت این دو در زمان که فقط مولفه ای اول کامرانی
باقی مانده باشد کاملاً مشخص است. تصریر حالت استاندار کل تصریر
مشترک است اما در حالت قضایی کل تصریر ^{این رنگ} خواهد بود.

سوال ۲

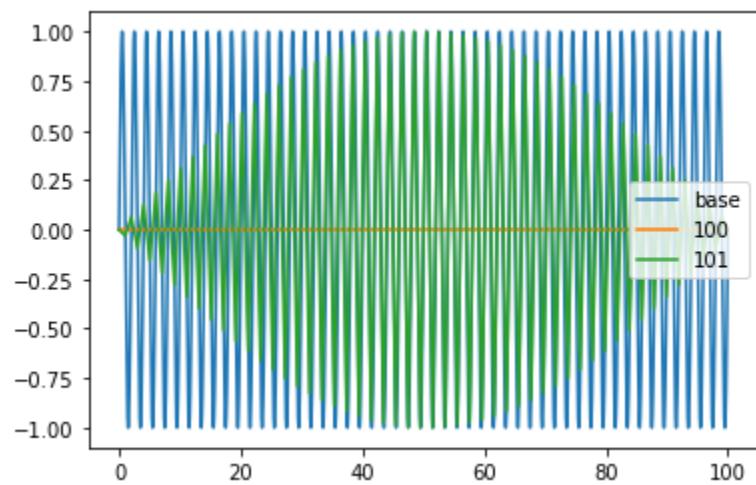
الف

براساس تکه که مشتبه صورت سوال:

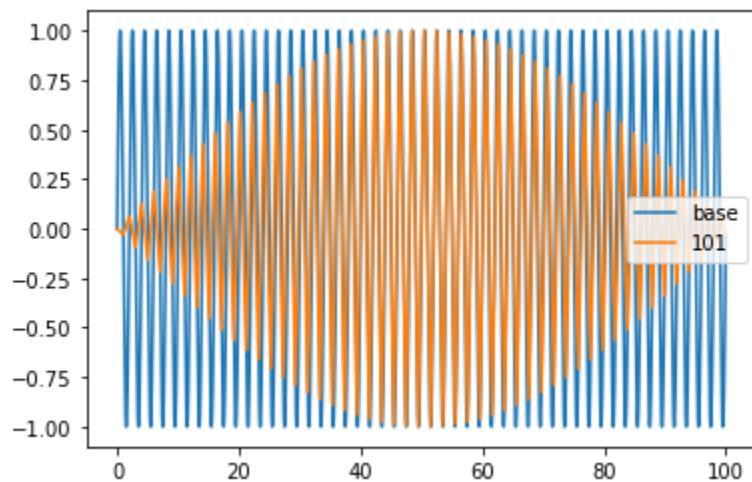
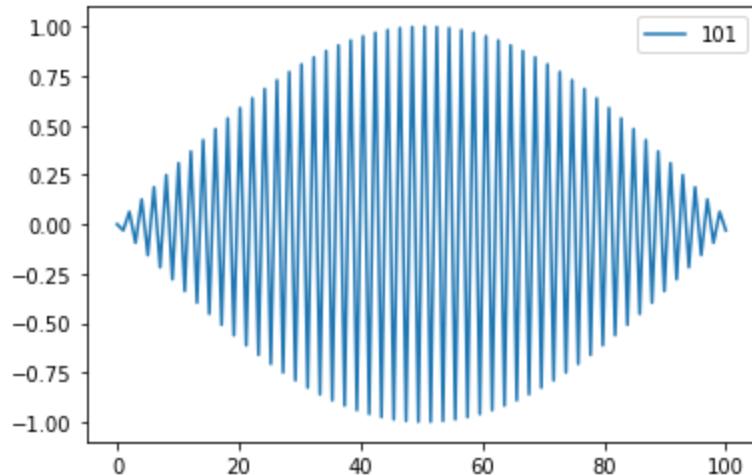


ب و ج

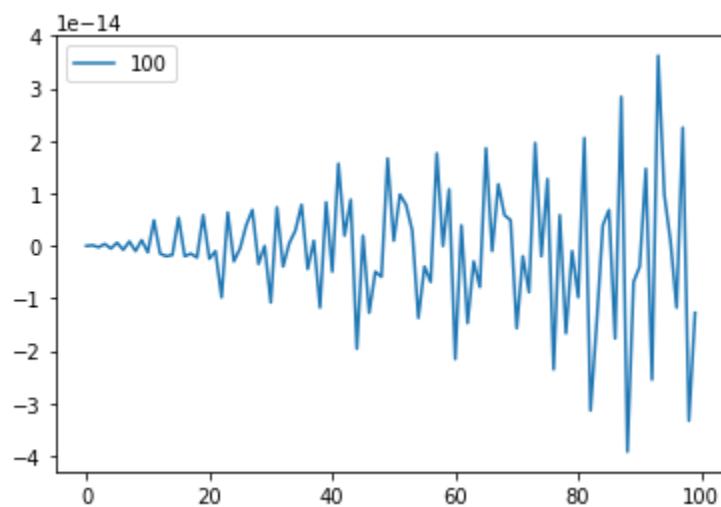
ترکیب سه حالت پایه و نمونه برداری ۱۰۰ و ۱۰۱ توجه شود ۱۰۰ هم نوسان دارد که در عکس جدا مشخص می‌شود.



: ۱۰۰ پلات



: ۱۰۱



توجه شود پیاده سازی ها از کدی به صورت زیر پیروی میکند:

```
x100 = [x[i] for i in range(0,len(x),100)]
```

```
y100 = [y[i] for i in range(0,len(x),100)]
```

```
x101 = [x[i] for i in range(0,len(x),101)]
```

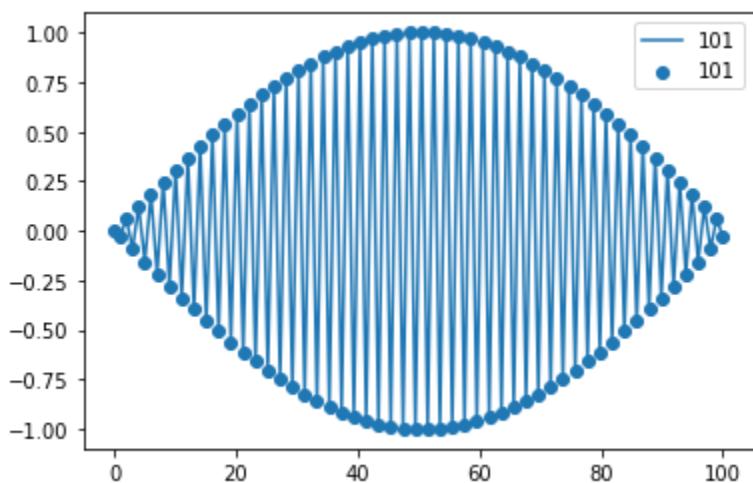
```
y101 = [y[i] for i in range(0,len(x),101)]
```

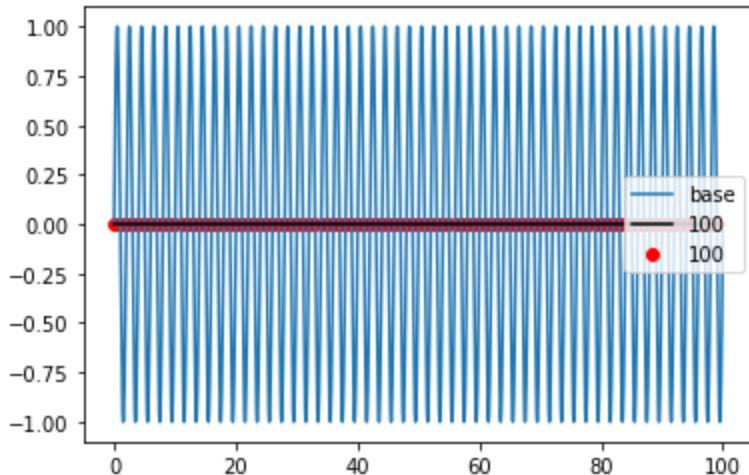
پلات هم مثلا داریم:

```
plt.plot(x,y,label='base')
plt.plot(x100,y100,label='100')
plt.plot(x101,y101,label='101')
plt.legend()
plt.show()
```

۵

در راستای پاسخ دهی به این بخش پلات با نمایش نقاط سمیل آورده شده است:





برای شرح اینکه چه اتفاقی می‌افتد یک توضیح مفهومی بدهیم. اگر نمونه برداری طبق قانون نایکوئیست باشد تمام قله ها دره ها مشخص می‌گردند و یک سیگنال مطلوب باز یابی می‌شود. اما در مورد نمونه برداری 10^1 هر بار به اندازه 10^1 بیشتر جلو می‌رود و مثلاً نقطه ای که باید در 10^1 قرار می‌گرفت اینبار کم بیشتر می‌گردد و این روند ادامه دارد تا زمانی که 10^1 بار این 10^1 ها رخ دهند تا یک نوسان کامل جلو بیفتد و در نتیجه نمونه خروجی در حالی که رفتار نوسان شدید سیگنال اولیه را دارد اما هر 10^1 نمونه یک دوره آن خواهد شد پس مانند شکل اول این بخش می‌شود. در مورد نمونه برداری 10^0 مشکل این است که نمیتواند از قله ها و دره ها نمونه برداری کند و فقط در نقاط صفر با در نظر گرفتن خطای ممکن نمونه برداری می‌شود پس وقتی در کنار سیگنال اولیه رسم می‌شوند مانند این است که همه روی یک خط با ارتفاع α هستند و خروجی نیز خط مانند است. البته به علت خطای ممکن و امکان رخداد کمی نویز یا ... خط در ظاهر در صورتی که با دقیق بسیار زیاد درشت نمایی زیاد مشاهده گردد رفتاری تصادفی مانند ایجاد خواهد کرد.

۵

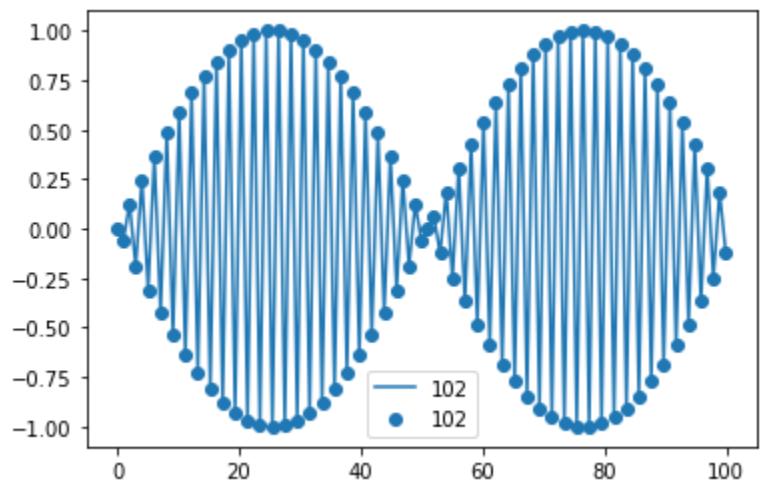
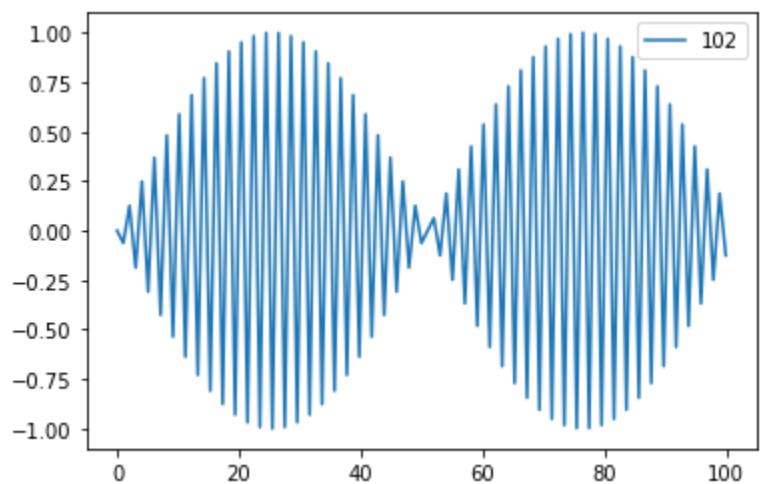
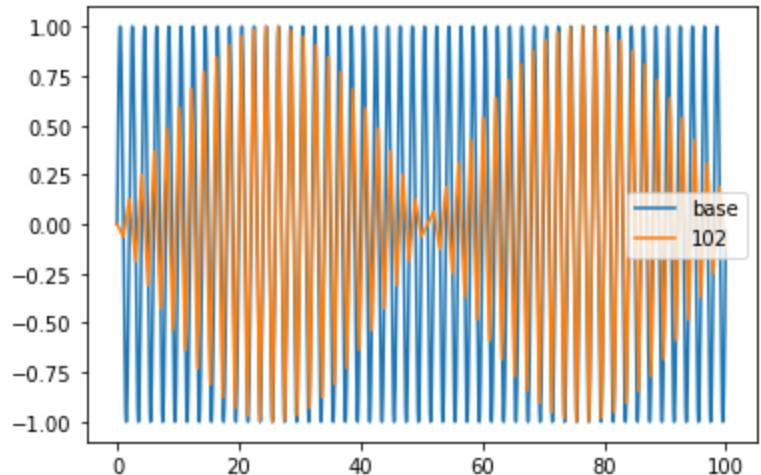
درسه تصویر زیر نمونه برای 10^2 آورده شده است. سپس برای بازه بیشتری از اعداد نیز رسم شده است. به طور کلا اگر نرخ نمونه برداری با 10^0 که همان خط یابنقطاط صفر بود اگر آلفا اختلاف داشته باشد پس در 10^0 تقسیم بر آلفا دور که برود یک دوره کامل می‌گردد. پس تعداد دوره های قابل مشاهده نیز :

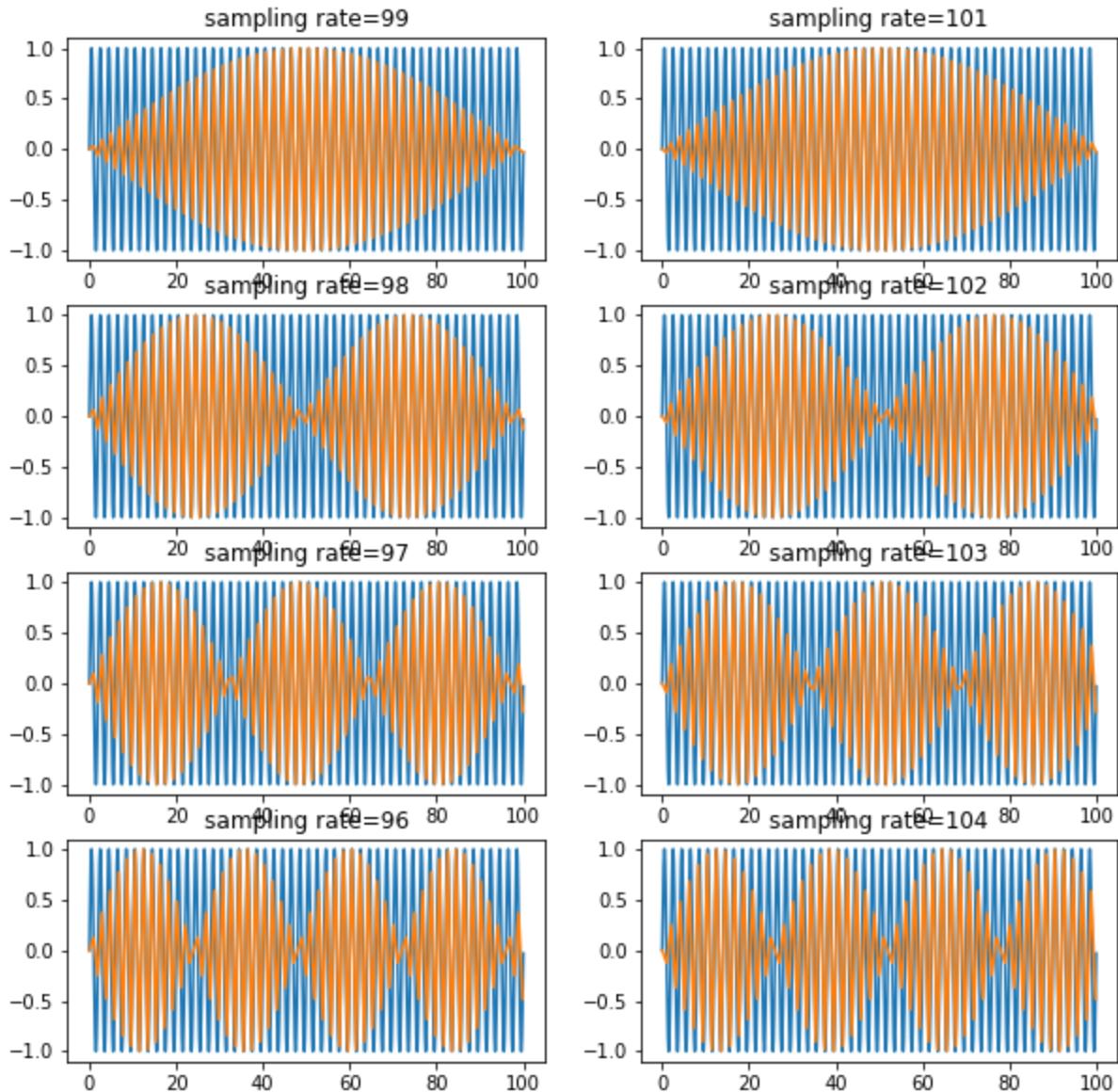
$$\text{Alpha} = \text{abs}(n-100)$$

$$\text{Period} = 100/\text{alpha}$$

$$\text{Number of wave} = 100/(100/\text{alpha}) = \text{alpha}$$

و مطابق انتظار مثلا در 10^2 که آلفا 23 می‌شود 2 بار قله موج را میبینیم.





سوال ۳

الف

با سرچ sRGB در گوگل و طول موج های آن، اعدادی بدست می اید که فایل متند میتوان حدس زد کدام یک از کانال ها مناسب تر هستند.

```
# based on Landa.txt:
b_channel = 20
g_channel = 30
r_channel = 52
```

در ادامه با استخراج شماره کانال‌ها، نیاز به نرمال‌سازی است. توجه شود تابعی که در ادامه مشاهده می‌فرمایید به نظر بهینه نیست اما در کمال تعجب عملیات رایج بر روی نامپای جواب نمیداد.

```
def img_normalizer_(origin_a):
    a = np.copy(origin_a)
    mi = min(map(min,a))
    ma = max(map(max,a))
    while(mi<-2000 or ma>20000):
        for i in range(len(a)):
            for j in range(len(a[0])):
                a[i][j] = a[i][j]*0.5
        mi = min(map(min,a))
        ma = max(map(max,a))

    mi = min(map(min,a))
    ma = max(map(max,a))
    while(mi<0 or ma>255):
        for i in range(len(a)):
            for j in range(len(a[0])):
                a[i][j] = (a[i][j] - mi)/(ma-mi)*255
        mi = min(map(min,a))
        ma = max(map(max,a))

    print(mi,ma)
    return a
```

در تابع فوق ذکر چند نکته حائز اهمیت است. اول اینکه در ابتدا همه‌ی مقادیر را تا زمانی که در رنج معینی قرار نگیرند تقسیم بر دو میکنیم چرا که وارینگ اورفلو شدن پدیدار می‌شود. سپس هم از فرمول مرسوم (اختلاف کمینه و بیشینه)/(داده منهای کمینه) برای نرمال‌سازی استفاده شد.

جهت بهبود کیفیت تصویر رنگ‌های مختلف در ضرایب مختلف ضرب شدن. در نهایت خروجی حاصله:



اما اگر براساس RGB معمولی کانال ها انتخاب میشندند:



همچنین در گام بعد تصمیم گرفته شد که در تابع نرمال ساز مقادیر منفی حذف گردند و اتفاقاً خروجی مطلوبی حاصل شد. جهت افزایش کیفیت و روشنایی همه کانال ها در ۲.۵ ضرب شده اند:



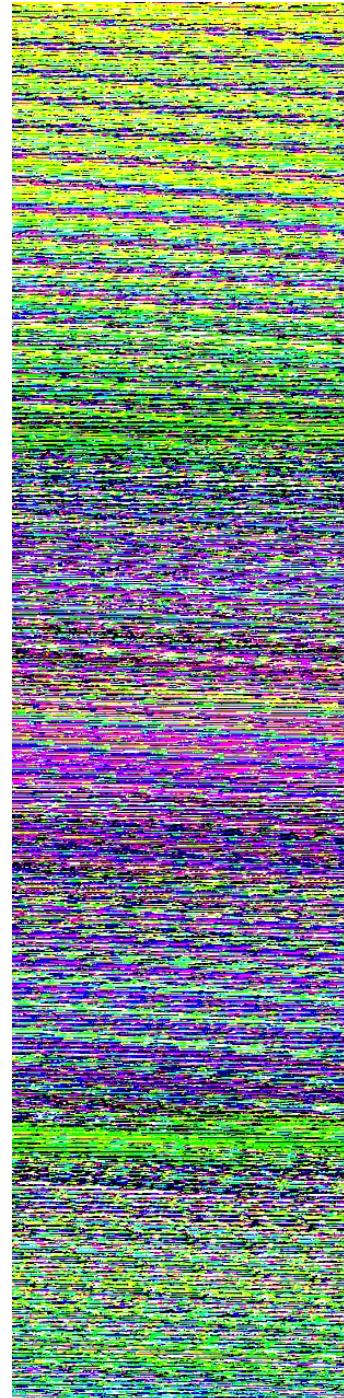
ب

در این قسمت از

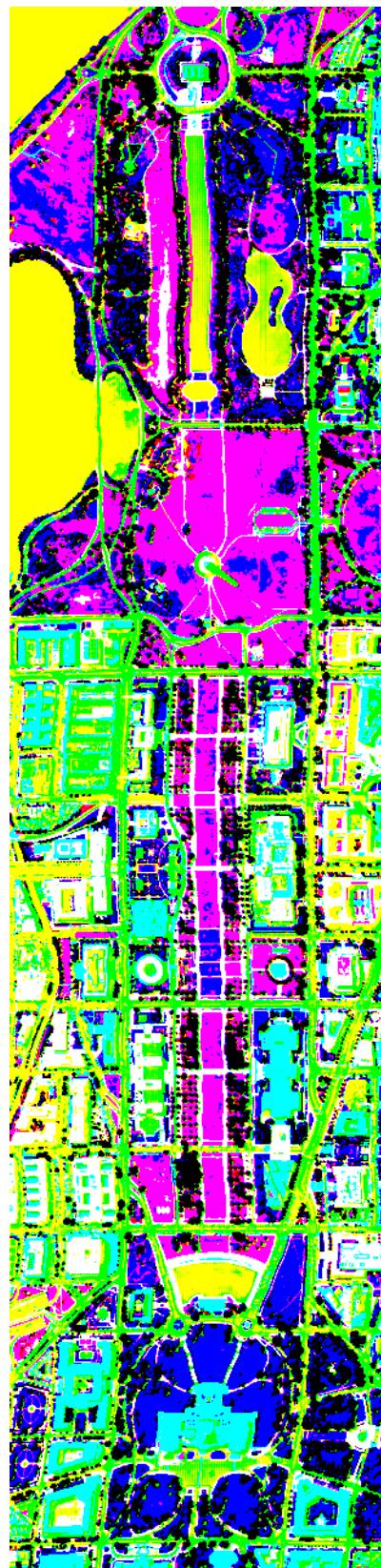
```
pca = PCA(n_components=n)
```

استفاده شد که n به ترتیب ۳ و ۱ مقدار دهی شد.

همچنین یک چالش زمانی بود که نیاز به `reshape` داشتیم و شکل بهم میریخت پس به وضوح ماتریس ترانسپوز استفاده گردید.
بدون استفاده از آن چنین شکلی حاصل نمیشد که مسلماً مطلوب نیست:



در ادامه نمونه صحیح با اعمال مناسب ترنسپوز را مشاهد می فرمایید:



همچنین برای داده فوق اگر سه کانال جدا رسم شوند داریم:



در نمونه تک کاناله نیز خروجی به شکل زیر در آمده است:



ج

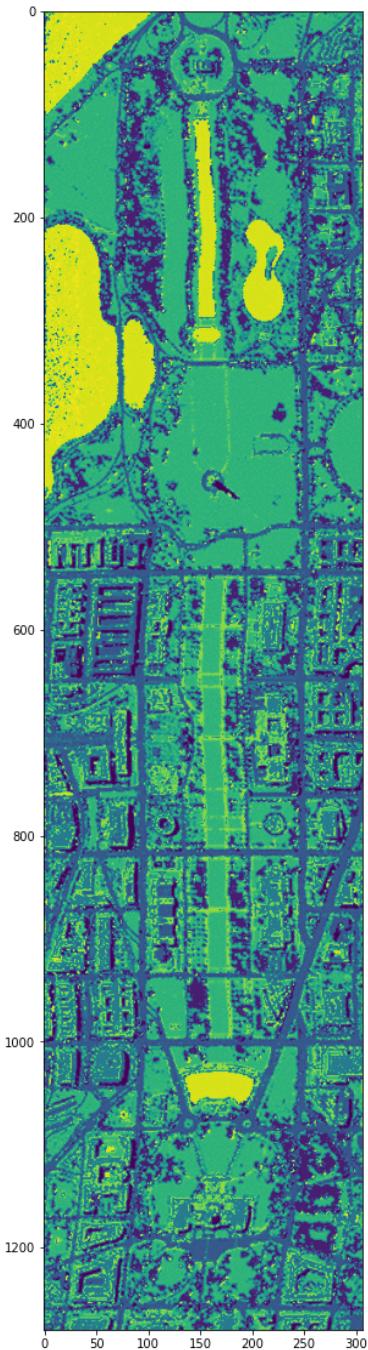
در این بخش برچسب داده‌ها در فایل `gt` هست و داده‌های اصلی هم که همان ۱۹۱ کانال بخش قبلی هستند.
برای نمایش داده `gt` با `cv2_show` تصویر سیاه سفید زیر حاصل می‌شود:



اما به کمک

```
gt = imageio.imread('GT.tif')
f = plt.figure()
f.set_figwidth(40)
f.set_figheight(20)
plt.imshow(gt)
```

به تصویر زیر دست می‌یابیم:



سپس داده‌ها را reshape می‌کنیم و با کمک:

```
from sklearn.discriminant_analysis import  
LinearDiscriminantAnalysis as LDA
```

و پیاده سازی پایین نتیجه بدست می اید.

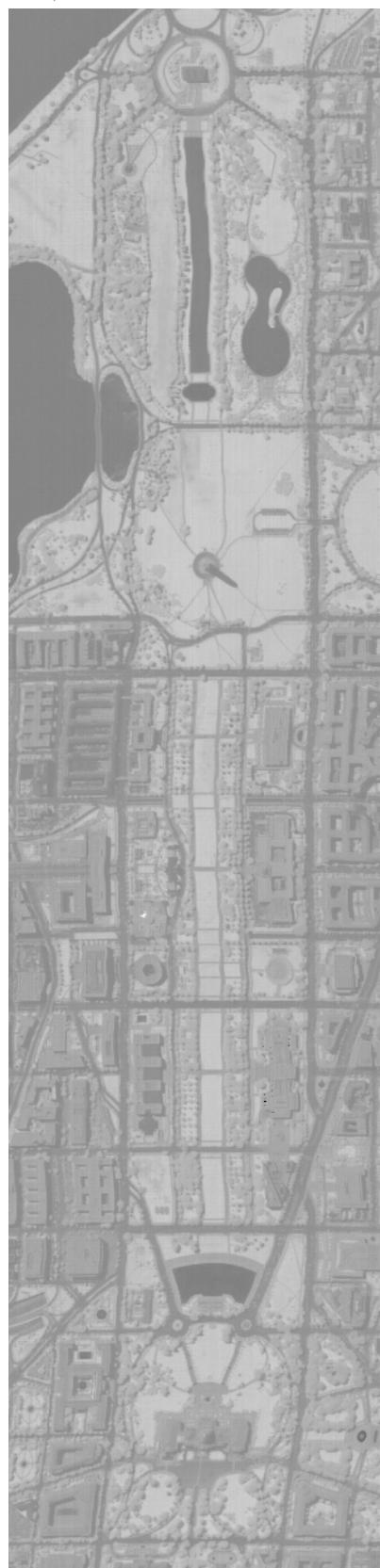
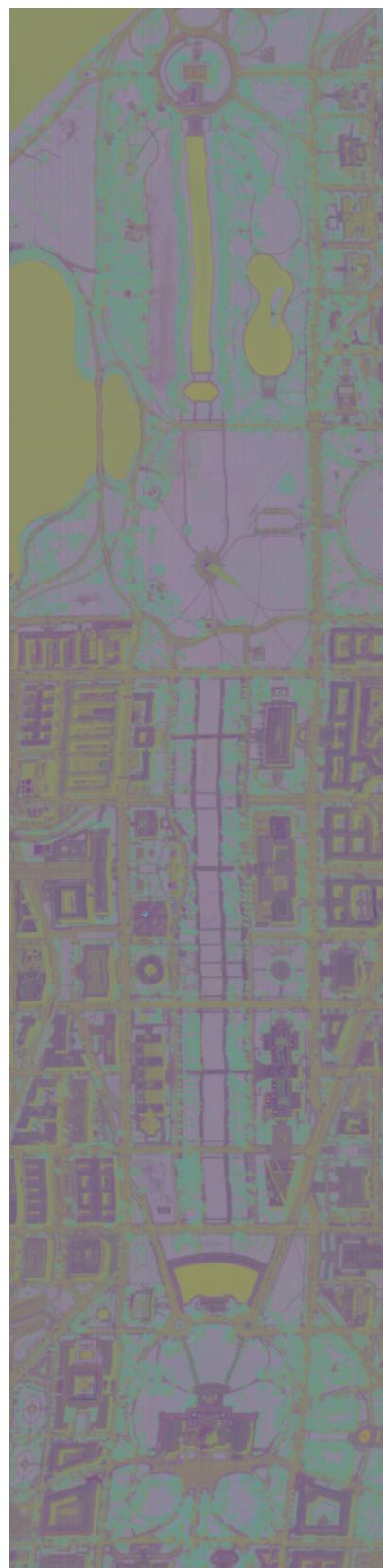
در این راستا تابع نرمال سازی:

```
def img_normalizer(x,max_val=255):  
    shape_ = x.shape  
    print(shape_)  
    x = x.reshape(-1)  
    mi = min(x)  
    ma = max(x)  
    x = (x-mi)/(ma-mi)*max_val  
  
    x = x.reshape(shape_)  
    return x
```

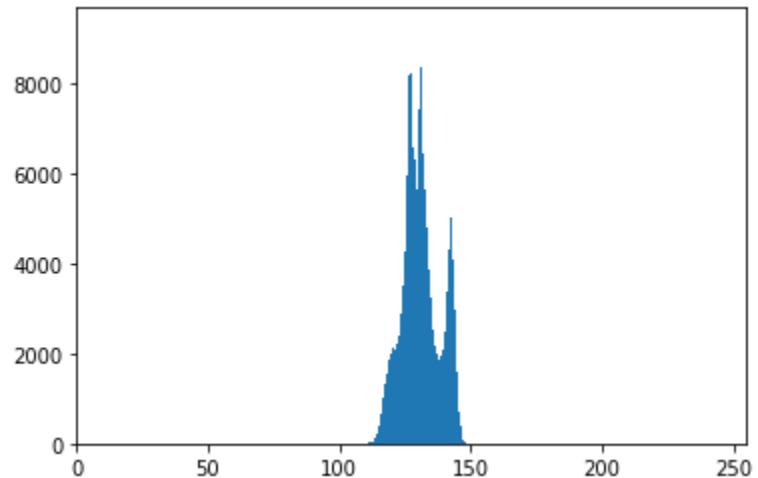
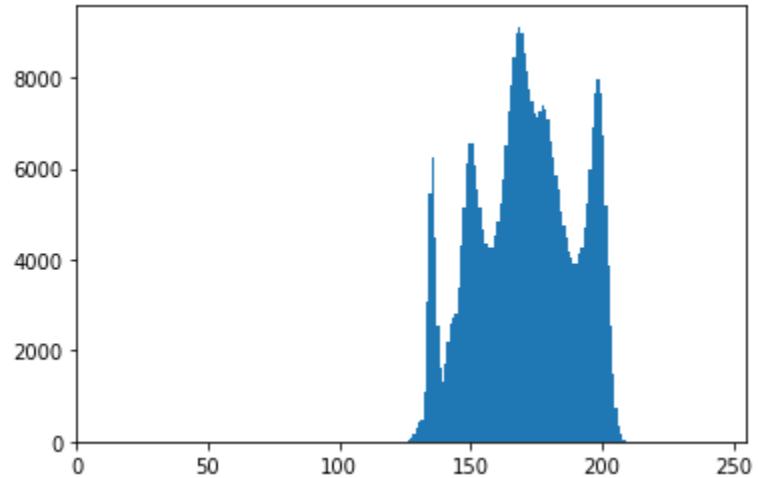
و تابع LDA آماده شده:

```
def LDA_function(n_components,x,y):  
    lda = LDA(n_components=n_components)  
    lda.fit_transform(x, y)  
    out = lda.transform(x)  
    out = out.reshape((1280,307,n_components))  
    normal_out = img_normalizer(out)  
    return normal_out
```

برای ۱ و ۳ کامپوننت به ترتیب داریم:



اما به وضوح یک مشکل وجود دارد به این صورت که تصاویر فوق به نظر میرسد رنج رنگی مطلوبی ندارند و هیستوگرام سیاه/سفید و رنگی به این صورت است:



با نادیده گرفتن محدود داده های پرت که در نزدیکی ۰ یا ۲۵۵ هستند و انتخاب یک رنج محدود تر مثلا در شکل اول ۱۱۵ و ۱۴۵ را گسترده سازی این رنج توسط تابع:

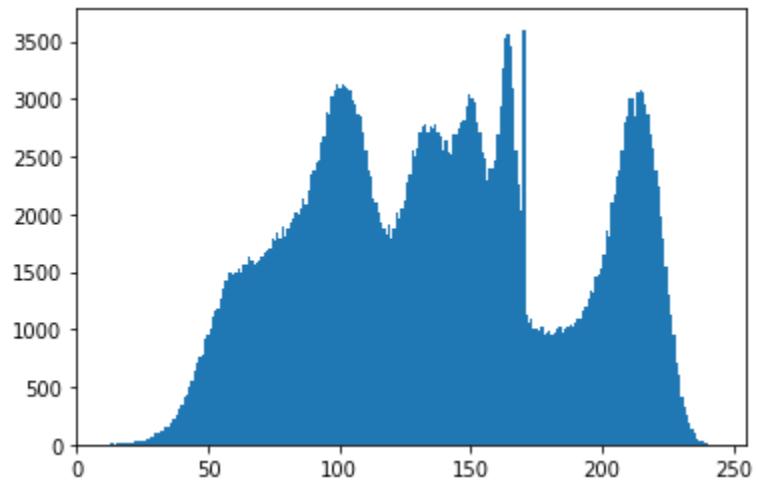
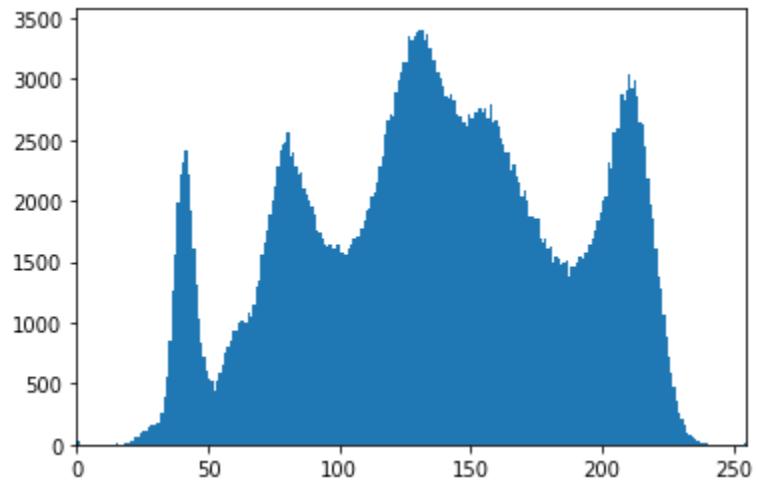
```
def rerange(range_ , img):
    s = img.shape
    img = img.reshape(-1)
    for i in tqdm(range(len(img))):
        img[i] = min(range_[1],max(range_[0],img[i]))
    img = img_normalizer(img)
    img = img.reshape(s)
    return img
```

و همچنین کد رسم هیستوگرام:

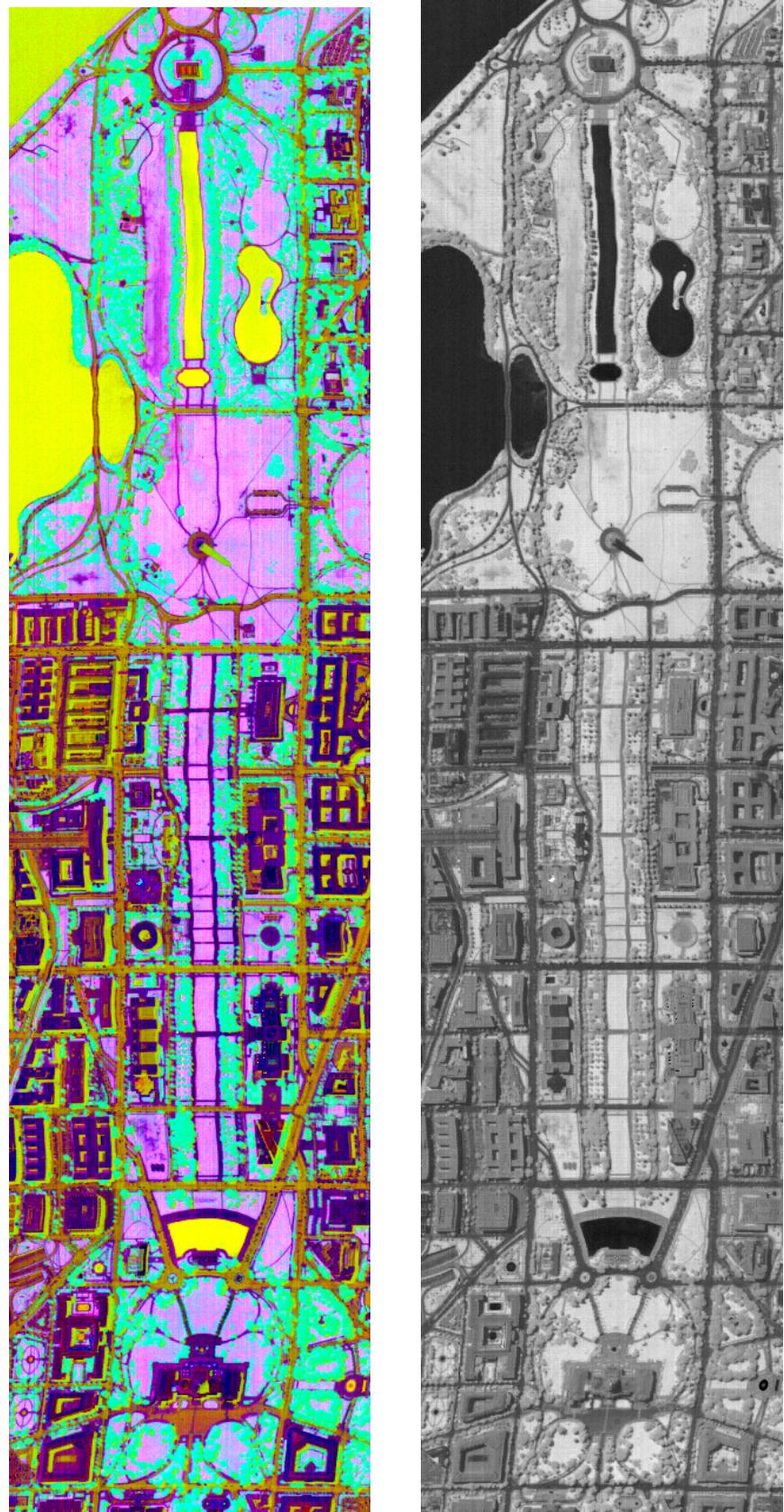
```
def draw_histogram(x):
```

```
vals = x.mean(axis=2).flatten()  
# plot histogram with 255 bins  
b, bins, patches = plt.hist(vals, 255)  
plt.xlim([0,255])  
plt.show()
```

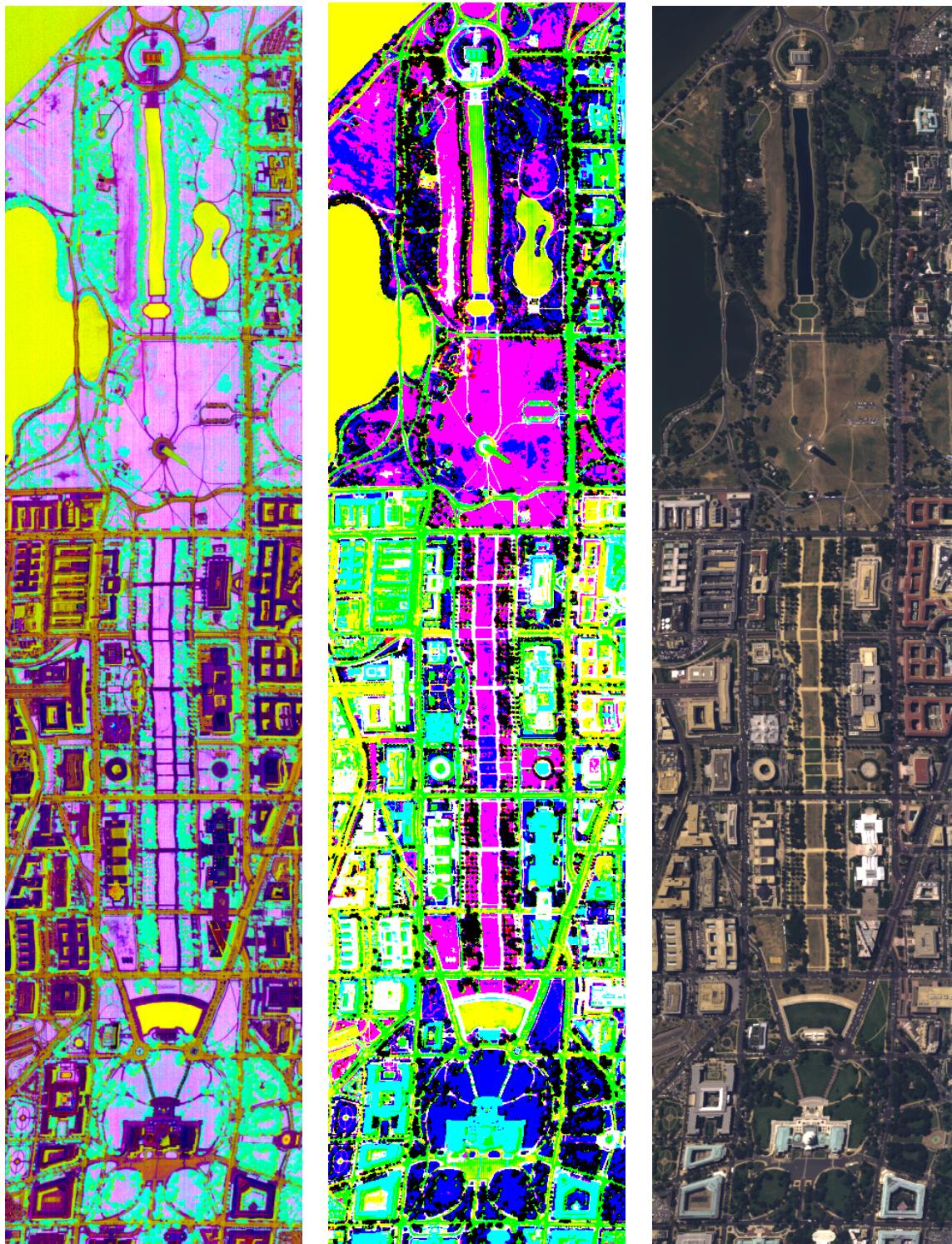
به دو هیستوگرام زیر دست می یابیم:



و تصاویر خروجی به شکل زیر خواهد بود که وضوح به مراتب بیشتری دارند:



خروجی‌های حاصل شده در بخش‌های الف و ب و ج ۳ کاناله داریم:



با مقایسه نمونه‌های بالا خوب نمونه RGB که برای چشم ما اشناتر است به وضوح قابل تشخیص تر است اما در صورتی که فرد بخواهد بدون دقت در جزئیات متوجه تفاوت بافت شود دو گزینه دیگر متمایز ترند و در این بین از نظر من LDA بهتر بوده است چراکه مثلا تشخیص پوشش گیاهی و ساختمان‌ها راحت تر است. البته چندان دور از ذهن هم نیست چراکه از gt به عنوان داده بیشتر بهره می‌برد.