نام و نام خانوادگی: **امیرحسین علی‌بخشی**
شماره‌ی دانشجویی: **۹۷۳۱۰۹۶**

پروژه پایان ترم جبر خطی

استاد: **دکتر شجاعی**
درس: **جبر خطی کاربردی**

فهرست مطالب

نام و نام خانوادگی: **امیرحسین علی‌بخشی**
شماره‌ی دانشجویی: **۹۷۳۱۰۹۶**

پروژه پایان ترم جبر خطی

استاد: **دکتر شجاعی**
درس: **جبر خطی کاربردی**

چکیده

یکی از کاربرد های مهم ماتریس ها، ذخیره کردن اطلاعات یک سری داده (بعنوان مثال یک عکس) در ماتریس است. حال اگر بخواهیم تغییراتی بر روی عکس اعمال کنیم مستقیماً به سراغ عکس نمیرویم، بلکه تمام اعمال مورد نظر را بر روی ماتریس دارای اطلاعات عکس اصلی انجام میدهیم و در صورت نیاز دوباره ماتریس را به عکسی جدید تبدیل میکنیم. این گزارش در واقع یک نمونه از کاربرد های فراوانی است که میتوان به کمک ماتریس ها بر روی عکس ها تغییرات اعمال کرد.

مقدمه

برای انجام این پروژه از زبان برنامه نویسی python استفاده میکنیم. برای این منظور به دو کتابخانه‌ی این زبان احتیاج خواهیم داشت:

- **numpy**: برای کار با ماتریس ها و کاربردهای آن (نظیر ضرب ماتریس ها، بدست آوردن مقادیر و بردار های ویژه و ...) از این کتابخانه استفاده میکنیم.
- **PIL**: برای کار با عکس ها در پایتون از کتابخانه‌ی PIL که در واقع مخفف Python Imaging Library میباشد استفاده میکنیم. میتوان با این کتابخانه به کمک اسم یک عکس آن را وارد برنامه کرد و روی آن عملیات مختلفی انجام داد.(تغییر اندازه‌ی عکس، نمایش عکس به کمک برنامه‌ای که نوشته‌ایم و ...)

```
import numpy as np
from PIL import Image
```

به کمک این دو کتابخانه، میتوان عکس را به ماتریس تبدیل کنیم و پس از انجام عملیات روی پیکسل های آن، دوباره ماتریس حاصل را به یک عکس جدید تبدیل کنیم. سپس به کمک عبارت زیر عکس را به ماتریسی ۳×۴۰۰۰۰۰۰ تبدیل میکنیم.

برای قسمت سوم که مربوط به ترسیم ابرداده ها میباشد،علاوه بر این دو، باید کتاب خانه ی زیر را نیز import کنیم(که در خود کتابخانه ی PIL قرار دارد):

```
from PIL.ExifTags import TAGS
```

که در بخش **ترسیم ابرداده ها** در مورد آن صحبت خواهد شد.

مراحل کار

انتخاب عکس(Q1)

در ابتدای برنامه با صفحه ی زیر مواجه خواهیم شد:

```
Hi! How do you want to work with this
program?
 1 - using ready image
 2 - using another image
-1 - exit the program
please enter your choice:
```

با انتخاب ۱ و ۲ نوع ورودی دادن به برنامه مشخص خواهد شد بعد از انجام عملیات روی عکس (در صورت وجود در پوشه files که خود فایل پروژه هم در آن قرار دارد) تا زمانی که ورودی ۱- را به برنامه ندهیم برنامه اجرا خواهد شد و عکس ها را بصورت متوالی از ما دریافت خواهد کرد. عکسی که به صورت پیشفرض برای برنامه در نظر گرفته شده است عکس زیر میباشد:

ذخیره عکس بصورت ماتریس(Q2)

پس از انتخاب عکس آن را به کمک resize به یک عکس ۲۰۰۰×۲۰۰۰ پیکسل تبدیل میکنیم(واضح است که اگر عکسی در ابتدا بصورت مستطیلی باشد، پس از این عملیات بصورت عمودی یا افقی فشرده خواهد شد تا بصورت مربعی تبدیل شود، مانند شکل زیر که فشرده شده ی عکس بالا میباشد).

سپس آرایه ای دو بعدی از عکس ورودی میسازیم(h و w برابر ۲۰۰۰ میباشند.):

```
image_matrix = np.array(image)
image_matrix.resize((h * w, 3))
```

در آرایه‌ی دو بعدی به دست آمده هر یک از ۴۰۰۰۰۰۰ سطر متعلق به یکی از پیکسل های عکس ورودی خواهند بود و هر کدام از این پیکسل ها سه مولفه خواهند داشت که میتوانند رنگ پیکسل را در RGB بیان کنند. مولفه های این پیکسل ها به ترتیب از چپ به راست Red، Green و Blue خواهند بود که هر یک مقداری بین ۰ تا ۲۵۵ دارند و به کمک آن ها میتوان تمام رنگ ها را تولید کرد و از کنار هم قرار دادن آنها عکس های گوناگونی ساخت. در **این** لینک اطلاعات بیشتر در خصوص رنگ های RGB توضیح داده شده است.

نکته ی مهم در *این* پروژه این است که ابعاد یک ماتریس و آرایه دقیقا برعکس هم هستند. برای همین موضوع برای نمایش ماتریس یه ضرب ماتریس ها ابتدا آنها را ترانهاده میکنیم که به فرم ماتریس شبیه شوند و سپس با آنها کار میکنیم. به بیان واضح‌تر، یک ماتریس mxn در برنامه ی ما یک آرایه‌ی nxm خواهد بود.

ماتریس

$$\begin{bmatrix} R_1 & R_2 & & R_n \\ G_1 & G_2 & \dots & G_n \\ B_1 & B_2 & & B_n \end{bmatrix}$$

آرایه‌ی دو بعدی

```
[[ R₁ G₁ B₁ ]
 [ R₂ G₂ B₂ ]
      ⋮
 [ Rₙ Gₙ Bₙ ]]
```

ترسیم ابرداده ها (Q3)

برای ترسیم ابرداده ها تابعی مانند زیر تعریف میکنیم:

```
def print_metadata(image):
    has_metadata = False
    exifdata = image.getexif()
    for tag_id in exifdata:
        tag = TAGS.get(tag_id, tag_id)
        data = exifdata.get(tag_id)
        # decode bytes
        if isinstance(data, bytes):
            has_metadata = True
            data = data.decode()
        print('   >>>',f"{tag:25}: {data}")
    return has_metadata
```

نکته‌ای که در این مورد لازم به ذکر میباشد این است که لزوما تمامی عکس ها دارای ابرداده نمیباشند. به عنوان مثال عکس پیش‌فرضی که در این گزارش استفاده شده است فاقد این اطلاعات است و معمولا عکس هایی ابرداده دارند که توسط دوربین گرفته شده اند. به عنوان مثال عکس زیر با دوربین موبایل گرفته شده است:

این عکسی است که دارای ابرداده میباشد. بنابراین با فراخوانیِ تابع print_metadata ، اطلاعات مربوط به این عکس به صورت زیر نمایش داده خواهد شد(این عکس هم در پوشه ی فایل قرار دارد و با در آوردن خط کد مربوط به آن (خط ۱۲۶) از حالت کامنت قابل دسترسی است):

```
>>> Q3: META DATAS :
   >>> ExifVersion              : 0220
   >>> ShutterSpeedValue        : (564, 100)
   >>> ApertureValue            : (252, 100)
   >>> DateTimeOriginal         : 2020:07:19 11:59:14
   >>> DateTimeDigitized        : 2020:07:19 11:59:14
   >>> BrightnessValue          : (282, 100)
   >>> ExposureBiasValue        : (0, 10)
   >>> MaxApertureValue         : (116, 100)
   >>> MeteringMode             : 3
   >>> Flash                    : 0
   >>> FlashPixVersion          : 0100
   >>> FocalLength              : (430, 100)
   >>> UserComment              :
   >>> ColorSpace               : 1
   >>> ComponentsConfiguration  :
```

```
>>> ExifImageWidth           : 4032
>>> SubsecTime               : 0756
>>> SubsecTimeOriginal       : 0756
>>> SubsecTimeDigitized      : 0756
>>> ExifImageHeight          : 1960
>>> ImageLength              : 1960
>>> Make                     : samsung
>>> Model                    : SM-N960F
>>> Orientation              : 1
>>> YCbCrPositioning         : 1
>>> ExposureTime             : (1, 50)
>>> ExifInteroperabilityOffset: 815
>>> XResolution              : (72, 1)
>>> FNumber                  : (240, 100)
>>> SceneType                :
>>> YResolution              : (72, 1)
>>> ImageUniqueID            : J12LLKL00SM
>>> ExposureProgram          : 2
>>> CustomRendered           : 0
>>> ISOSpeedRatings          : 100
>>> ResolutionUnit           : 2
>>> ExposureMode             : 0
>>> ImageWidth               : 4032
>>> WhiteBalance             : 0
>>> Software                 : N960FXXU5ETF5
>>> DateTime                 : 2020:07:19 11:59:14
>>> DigitalZoomRatio         : (0, 0)
>>> FocalLengthIn35mmFilm    : 26
>>> SceneCaptureType         : 0
>>> Contrast                 : 0
>>> Saturation               : 0
>>> Sharpness                : 0
>>> ExifOffset               : 225
```

ولی در مورد عکس اصلی پیام METADATA NOT FOUND! مشاهده خواهد شد.

محاسبه‌ی میانگین (Q4)

با توجه به فرمول زیر میانگین را باید محاسبه کنیم و با کمک آن تابع زیر را تعریف میکنیم:

$$M = \frac{1}{4000000}(x_1 + \cdots + x_{4000000})$$

```python
def calculate_mean(img):
    n = img.shape[0]
    mean =[[.0 , .0 , .0]]
    for pixel in range(img.shape[0]):
        mean = mean + img[pixel]
    mean = mean / n
    return mean
```

که خروجی این تابع بردار میانگین خواهد بود.

محاسبه‌ی ماتریس covariance (Q5)

با رابطه ی $S = \frac{1}{3999999} BB^T$ میتوان ماتریس کوواریانس را محاسبه کرد که در آن $B = [(x_1 - M) \dots (x_{4000000} - M)]$

برای همین موضـــوع تابع calculate_covariance_matrix(img, m) را تعریف میکنیم که عکس و میانگین را به عنوان آرگمان میگیرد و خروجی آن ماتریس کوواریانس یا همان S خواهد بود:

```
def calculate_covariance_matrix(img, m):
    n = img.shape[0]
    B = img
    for pixel in range(img.shape[0]):
        B[pixel] = B[pixel] - m
    B = B.transpose()
    s = (B.dot(B.transpose()))/(n-1)
    return s
```

فرم کلی ماتریس کوواریانس بصورت زیر میباشد:

$$S = \begin{bmatrix} s_{RR} & s_{RG} & s_{RB} \\ s_{GR} & s_{GG} & s_{GB} \\ s_{BR} & s_{BG} & s_{BB} \end{bmatrix}$$

که در واقع بیانگر رابطه میان رنگ های عکس مورد نظر میباشد.منطقی است که این ماتریس، ماتریسی متقارن خواهد بود.($S_{ij} = S_{ji}$)

واریانس و هم‌بستگی داده ها (Q6)

واریانس در این مبحث، همان درایه های روی قطر اصلی ماتریس کوواریانس (s) خواهند بود. که به کمک تابع ریز آن را میسازیم:

```
def calculate_variance_matrix(s):
    variance = np.array([[.0 , .0 , .0]])
    for v in range(3):
        variance[0][v] = s[v][v]
    return variance
```

برای بررسی هم بستگی دو داده باید به درایه ی نظیر به سطر و ستون مربوط به آن دو داده در ماتریس کوواریانس نگاه کنیم. این مقدار در اصل همان کوواریانس مربوط به دو متغیر خواهد بود. اگر کوواریانس دو متغیر ۰ باشد میگوییم آن دو متغیر ناهمبسته یا uncorrelated خواهند بود. بطور کلی با در نظر گرفتن ماتریس کوواریانس میتوان گفت:

$$s_{ij} = \begin{cases} var(Ci) & if:\ i = j \\ cov(Ci, Cj) & if:\ i \neq j \end{cases}, \quad s_{ij} = \begin{cases} 0 \Rightarrow Ci\ and\ Cj\ are\ uncorrelated \\ otherwise \Rightarrow Ci\ and\ Cj\ are\ correlated \end{cases}$$

این قسمت در پروژه تحت عنوان COVARIANCE MATRIX ANALYSIS آورده شده که در آن ابتدا کوواریانس دو به دوی رنگ ها نشـان داده میشـوند و در صـورت ۰ بودن این مقدار در جلوی آنها عبارت xi and xj are uncorrelated. نمایش داده میشـود. سپس واریانس ها نشان داده میشوند. برای مثال در زیر ماتریس کوواریانس و آنالیز آن برای یک عکس ورودی دلخواه نمایش داده شده است:

```
>>> Q5: CALCULATING COVARIANCE MATRIX:

  [[ 0.0038     0.0057      0.0021      ]
   [ 0.0057     0.0042      0.0035      ]
   [ 0.0021     0.0035      0.004       ]]

----------------------------------------------------------------------
>>> Q6: COVARIANCE MATRIX ANALYSIS:
   >>> COVARIANCES ANALYSIS:
      >>> COV( X 1 , X 2 )= 0.0          ->      X 1 and X 2 are UNCORRELATED.
      >>> COV( X 1 , X 3 )= 0.0          ->      X 1 and X 3 are UNCORRELATED.
      >>> COV( X 2 , X 1 )= 0.0          ->      X 2 and X 1 are UNCORRELATED.
      >>> COV( X 2 , X 3 )= 0.0001
      >>> COV( X 3 , X 1 )= 0.0          ->      X 3 and X 1 are UNCORRELATED.
      >>> COV( X 3 , X 2 )= 0.0001
   >>> VARIANCES ANALYSIS:
      >>> VAR( X 1 )= 0.0
      >>> VAR( X 2 )= 0.0
      >>> VAR( X 3 )= 0.0001
```

کاهش بُعد (Q7)

ابتدا دو تابع تعریف میکنیم که مقدار ویژه ها و بردار های ویژه ی S را برایمان محاسبه کنند. به کمک مقدار های ویژه ماتریس قطری D را تشکلی میدهیم.

حال باید واریانس کل را حساب کنیم که برابر حاصل جمع درایه های روی قطر اصلی D میباشد:

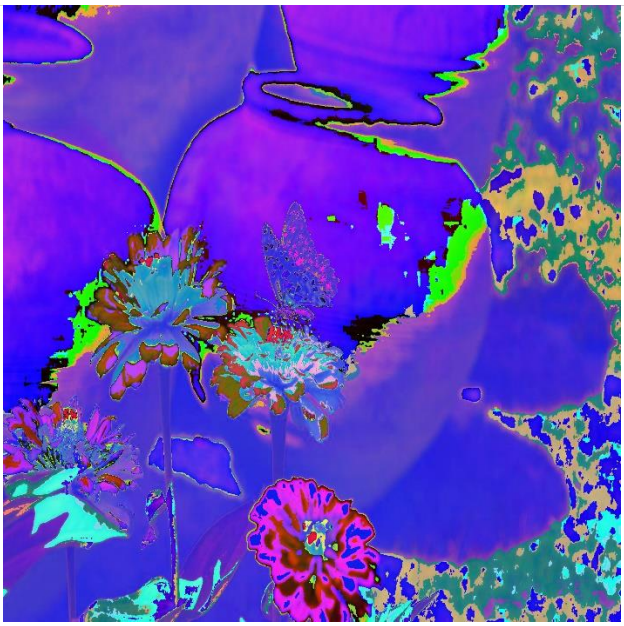
```python
def calculate_total_variance(variance_matrix):
    total_variance = .0
    for v in range(3):
        total_variance = total_variance + variance_matrix[0][v]
    return round(total_variance,2)
```

لازم به ذکر است که tr(D)=tr(S).

حال حاصل تقسیم مقادیر ویژه را بر واریانس کل محاسبه میکنیم و عبارت حاصل را بر حسب درصد بیان میکنیم. ۳ عبارت حاصل بیانگر سهم هر یک از سه رنگ RGB در واریانس کل میباشد.

تولید عکس جدید به کمک Analyze Component Principle (Q8)

ماتریس P را به کمک بردار ویژه های ماتریس S تولید میکنیم. و معکوس آن را از سمت چپ در ماتریس عکسمان ضرب میکنیم تا ماتریسی جدید به نام Y حاصل شود. ($Y = P^{-1}X$) با توجه ۳×۳ بودن P میتوان گفت ابعاد Y شبیه X است و همین یعنی میتوان آن را به عکسی با ابعاد ۲۰۰۰×۲۰۰۰ پیکسل تبدیل کرد. پس از ساخته شدن عکس از روی آرایه، برنامه آن را ابتدا با نام result.png در همان پوشه‌ی اصلی برنامه ذخیره میکند و سپس آن را برای ما به نمایش میگذارد:



نتیجه

همانگونه که در ابتدای گزارش گفته شـــد، میتوان به کمک ماتریس هایی که از داده های ما ســـاخته شـــده اند و همچنین به کمک اعمالی که میتوان روی ماتریس ها انجام داد، داده ها را بررسی کرد و حتی روی آنها تغییراتی انجام داد که ممکن است اطلاعات خیلی مفیدی در مورد عکس اولیه در اختیار ما قرار دهند؛ اطلاعاتی که مستقیما از عکس اولیه حاصل نمیشوند.