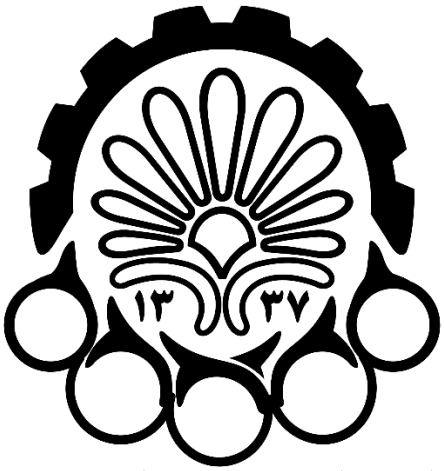


دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

تمر ین سوم درس جبرخطی کاربردی(عملی)

دکتر امیرمزلقانی

غلامرضا دار ۴۰۰۱۳۱۰۱۸

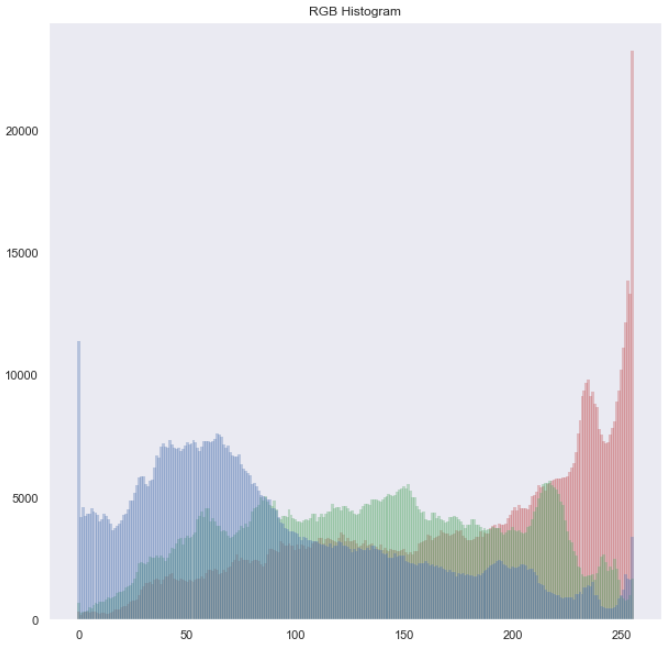
بهار ۱۴۰۱

فهرست مطالب

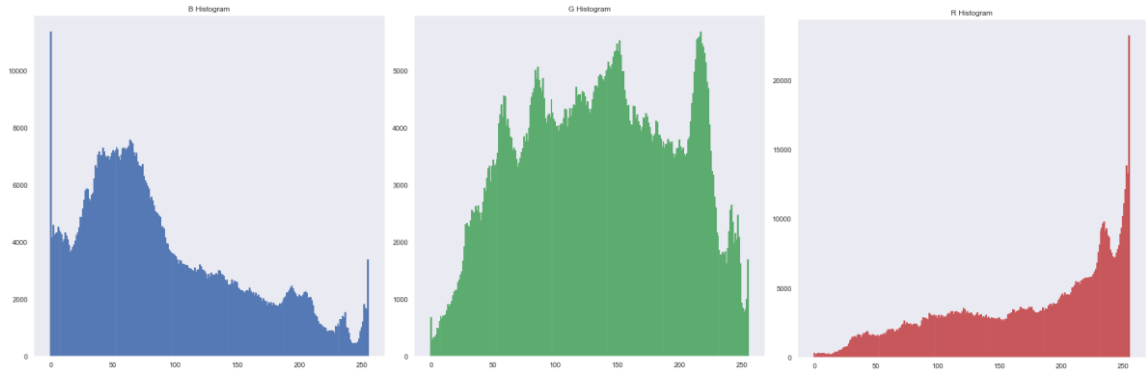
سوال ۱)

ابتدا تصویر را لود میکنیم و در یک آرایه نامپای ذخیره میکنیم.

سپس کانال‌های R,G,B تصویر را جدا میکنیم و در هر کدام به طور مستقل تعداد پیکسل‌های با مقادیر مختلف را می‌شماریم. سپس با استفاده از کتابخانه Matplotlib این اعداد را به عنوان ارتفاع نمودار ستونی رسم می‌کنیم.



در تصویر بالا، هیستوگرام های R,G,B را روی یکدیگر مشاهده میکنید. در ادامه این هیستوگرام ها را به طور مستقل برای هر کانال رسم میکنیم.



اگر به تصویر اصلی دقت کنید مشاهده میکنید که هیستوگرام مربوط به رنگ آبی در مقادیر کم فراوانی زیادی دارد. به این معنی که در سایه ها و بخش های تاریک تصویر رنگ آبی وجود دارد. از آنطرف، هیستوگرام رنگ قرمز در مقادیر بزرگ فراوانی زیادی دارد. اگر دقت کنید لباس کودکان و تعدادی از درخت‌ها که عموما روشن هستند قرمزرنگ هستند و در نواحی تیره، مانند سایه‌ها، رنگ قرمز دیده نمی‌شود. عمده رنگ سبز تصویر نیز مربوط به درختان می‌باشد که نه خیلی روشن هستند و نه خیلی تاریک که این را می‌توان در هیستوگرام مربوط به رنگ سبز مشاهده کرد.

سوال ۲)

ابتدا تصویر طوطی را فراخوانی میکنیم.

سپس نواحی سفید تصویر را جدا میکنیم تا طوطی از پس زمینه جدا شود.

در ادامه با استفاده از تبدیل داده شده در صورت سوال، تصویر بالا را دفورم می‌کنیم. نحوه اعمال تبدیل به این شکل است که ابتدا یک تصویر هم اندازه با تصویر اصلی تولید میکنیم. سپس هر پیکسل از تصویر جدید را برابر با رنگ پیکسل متناظر با تصویر بالا قرار می‌دهیم. اگر پیکسل i,j را به پیکسل i,j از تصویر اصلی نسبت دهیم، هیچ تغییری در تصویر ورودی ایجاد نمی‌شود.

اما اگر پیکسل i,j را به پیکسل $\begin{bmatrix} i_{new} \\ j_{new} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \lambda & 1 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix}$ نسبت دهیم، عمل Skew به تصویر سیاه و سفید اعمال می‌شود.

منتهی الان یک مشکل دیگر وجود دارد. همانطور که می‌دانیم مختصات ۰و۰ تصویر در بالا و سمت چپ آن قرار دارد. به همین دلیل عمل Skew یا هر تبدیل دیگری حول آن نقطه انجام می‌شود(از خواص تبدیل خطی این است که نقطه ۰و۰ دست نخورده باقی می‌ماند). اما این خاصیت برای ما مطلوب نیست و ما می‌خواهیم تمام این عملیات حول پای طوطی اتفاق بی‌افتد. برای این منظور ابتدا با آزمون و خطا مختصات پای طوطی را محاسبه می‌کنیم. (در تصویر زیر با رنگ قرمز مشخص شده است)

تغییر دیگری که نیاز است انجام دهیم این است که مختصات را از این شکل،

```
for i in range(parrot_shadow.shape[0]):
    for j in range(parrot_shadow.shape[1]):
```

به این شکل تغییر دهیم. با این کار هنگامی که i,j برابر ۰و۰ شوند، ما به پای طوطی رسیده ایم و تمام عملیات حول این نقطه رخ می‌دهند.

```
for i in range(-1*feet_y, parrot_shadow.shape[0]-feet_y):
    for j in range(-1*feet_x, parrot_shadow.shape[1]-feet_x):
```

از آنجایی که اعداد منفی را نمی‌توانیم به عنوان مختصات به تصاویر دهیم، مختصات i,j جدید را به صورت زیر با مختصات پای طوطی جمع میکنیم(T ماتریس تبدیل است).

```
i_p = int(T[0,0]*i + T[0,1]*j) + feet_y
j_p = int(T[1,0]*i + T[1,1]*j) + feet_x

i_new = i+feet_y
j_new = j+feet_x
```

در نهایت عمل زیر را برای هر پیکسل انجام می‌دهیم.

```
skewed_shadow[i_p, j_p] = parrot_shadow[i_new,j_new]
```

پس از انجام تغییرات گفته شده، تبدیل‌ها به درستی حول پای طوطی رخ می‌دهند.

در ادامه کافی است تصویر بدست آمده را در پیکسل‌هایی از تصویر اصلی قرار دهیم که طوطی وجود ندارد(قبلا محاسبه شده است). همچنین برای زیباترشدن تصویر نهایی، رنگ پیکسل‌های سایه طوطی را به رنگ روشن تری تغییر دادیم.

ایده دیگری که در این قسمت استفاده شد، بهره‌گیری از فاصله پیکسل‌ها از موقعیت پای طوطی بود. به این شکل که تصویری تحت عنوان falloff تعریف میکنیم و مقدار هر پیکسل آن برابر با فاصله آن پیکسل تا پای طوطی باشد.

پس از اعمال تصویر بالا به سایه محاسبه شده در مرحله قبل، به نتیجه های زیر می‌رسیم.

سوال ۳)

ابتدا با ساختار ماتریس Toeplitz آشنا می‌شویم.

$$A = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & \cdots & \cdots & a_{-(n-1)} \\ a_1 & a_0 & a_{-1} & \ddots & & \vdots \\ a_2 & a_1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & a_{-1} & a_{-2} \\ \vdots & & \ddots & a_1 & a_0 & a_{-1} \\ a_{n-1} & \cdots & \cdots & a_2 & a_1 & a_0 \end{bmatrix}$$

در ادامه دو بردار r,c تعریف میکنیم.

```
c = np.array([0, 1, 2, 3, 4,])
r = np.array([5, 6, 7, 8, 9,])
```

و با کمک دو حلقه ساده ماتریس روبرو را تولید میکنیم.

```
Toeplitz matrix:
[[0. 6. 7. 8.]
 [1. 0. 6. 7.]
 [2. 1. 0. 6.]
 [3. 2. 1. 0.]]
```

در مرحله بعد باید سعی کنیم این ماتریس را به شکل بالامثلثی در بیاوریم. برای پیاده سازی این قسمت از تکنیک‌های مشابه با تمرین اول(تبدیل ماتریس به فرم اشلون) استفاده کردیم. در ادامه به ماتریس بالامثلثی روبه‌رو می‌رسیم.

```
Upper triangular Toeplitz matrix:
[[3.         2.         1.         0.        ]
 [0.         6.         7.         8.        ]
 [0.         0.         6.44444444 7.88888889]
 [0.         0.         0.         6.78448276]]
```

درانتها باید دترمینان این ماتریس را محاسبه کنیم. می‌دانیم دترمینان ماتریس بالامثلثی برابر با حاصل‌ضرب درایه های ستون اصلی است.

```
Determinant: 787.0
```