

# گزارش پروژه پایانی درس سیستم‌های چندرسانه‌ای

نام و نام خانوادگی: امیرحسین علی‌بخشی

شماره دانشجویی: ۹۷۳۱۰۹۶

## بخش ۱: سوالات تشریحی

**Dithering: ۱ - ۱ چیست؟**

تکنیکی که بر اساس گسسته‌سازی (Quantization) عکس، آن را فشرده می‌کند.

۱ - ۲: دو مورد از الگوریتم‌های **Dithering** را نام برده و طرز کار آن‌ها را تشریح کنید.

### **Threshold Dithering:**

در این روش یک حد آستانه در نظر گرفته می‌شود و مقادیر تمام پیکسل‌های عکس با آن مقایسه می‌شود. اگر مقادیر بیشتر از حد آستانه باشد آن پیکسل سفید و در غیر این صورت سیاه می‌شود.

### **Ordered Dithering:**

در این روش مفهومی به نام پنجره لغزان تعریف می‌کنیم. این پنجره در اصل یک ماتریس (به آن Bayer Matrix می‌گوییم) می‌باشد که روی عکس قرار می‌گیرد و با مقادیر پیکسل‌های عکس مقایسه می‌شود. در صورتی که مقدار پیکسل از درایه‌ی متناظر این ماتریس بیشتر بود، آن پیکسل سفید و در غیر این صورت سیاه می‌شود.

۱ - ۳: در الگوریتم **Ordered Dithering** پنجره لغزان چه سایزهایی می‌تواند داشته باشد؟

از مقادیر زوج استفاده می‌کنیم.

۱ - ۴: تاثیر سایز پنجره لغزان در الگوریتم **Ordered Dithering** را با مثالی توضیح دهید.

تاثیر در دقت عکس می‌باشد. هرچه اندازه پنجره لغزان بزرگتر باشد، عکس فشرده شده به عکس اصلی نزدیک‌تر خواهد بود. در صفحه آخر این گزارش نمونه‌های مختلفی با اندازه پنجره‌های مختلف وجود دارد که این موضوع را تایید می‌کنند.

## بخش ۲: پیاده‌سازی

۲ - ۱: عکس با هر سایز دلخواه به عنوان ورودی گرفته شود.

برای این کار از کتابخانه‌های **numpy** و **PIL** استفاده می‌کنیم. به کمک این دو کتابخانه، پیکسل‌های عکس ورودی به صورت **RGB** در آرایه‌ای سه‌بعدی ذخیره می‌شوند:

```
original_image = Image.open(image_name)
image_matrix = np.array(original_image)
```

به عنوان مثال عکس ورودی ما برای این سوال عکس زیر خواهد بود:



۲ - ۲: تبدیل عکس به GrayScale.

همانطور که گفتیم عکس در آرایه سه‌بعدی ذخیره میشود. اگر ابعاد عکس ما  $m \times n$  باشد، ابعاد این آرایه  $3 \times m \times n$  خواهد بود. آن عدد سه هم مربوط به مقادیر RGB برای هر پیکسل خواهد بود. برای تبدیل عکس به GrayScale برای هر یک از پیکسل‌ها باید از سه مقدار RGB (که اعدادی بین ۰ تا ۲۵۵ میباشند) میانگین بگیریم و آن را داخل آرایه ای با ابعاد  $m \times n$  ذخیره کنیم.

$$GS_{ij} = \frac{R_{ij} + G_{ij} + B_{ij}}{3}$$

برای این منظور از کد زیر استفاده میکنیم:

```
grayscale_matrix = np.array([
    [
        np.average(image_matrix[i][j]) for j in range(image_size[1])
    ]
    for i in tqdm(range(image_size[0]), 'creating grayscale')
])

grayscale_image = Image.fromarray(grayscale_matrix)
```

پس از اعمال این کار روی عکس ورودی، با عکس زیر روبرو خواهیم شد:



۲ - ۳: اندازه پنجره لغزان به عنوان ورودی گرفته شود.

این مقدار در متغیری به نام `window` ذخیره میشود و از کاربر میخواهیم عددی وارد کند. این عددی که ما وارد کنیم عدد زوج باشد، ماتریس ما  $w \times w$  خواهد بود. اما اگر  $w$  فرد باشد، ماتریس ما  $(w-1) \times (w-1)$  خواهد بود.

۲ - ۴: تشکیل پنجره لغزان با اندازه وارد شده توسط کاربر.

شبه کد ساخت این پنجره لغزان به صورت رابطه بازگشتی زیر می‌باشد<sup>۱</sup>:

$$\mathbf{M}_{2n} = \frac{1}{(2n)^2} \times \begin{bmatrix} (2n)^2 \times \mathbf{M}_n & (2n)^2 \times \mathbf{M}_n + 2 \\ (2n)^2 \times \mathbf{M}_n + 3 & (2n)^2 \times \mathbf{M}_n + 1 \end{bmatrix}$$

از کد زیر برای پیاده‌سازی این تابع بازگشتی استفاده میکنیم:

```
def bayer(n):  
    if n == 1:  
        return np.array([[0,2],[3,1]])/((2*n)**2)  
    M = np.array(((2*n)**2)*bayer(n/2))  
    return np.concatenate((np.concatenate((M, M+2), axis=1), np.concatenate((M+3,  
        M+1), axis=1)), axis=0)/((2*n)**2)
```

چند نمونه از ماتریس‌های تولید شده به ازای اندازه‌های مختلف  $w$  به صورت زیر خواهد بود:

<sup>۱</sup> منبع: [https://en.wikipedia.org/wiki/Ordered\\_dithering](https://en.wikipedia.org/wiki/Ordered_dithering)

اندازه پنجره	ماتریس
$w = 2$	$\begin{pmatrix} 0 & 127.5 \\ 191.25 & 63.75 \end{pmatrix}$
$w = 4$	$\begin{pmatrix} 0 & 127.5 & 31.875 & 159.375 \\ 191.25 & 63.75 & 223.125 & 95.625 \\ 47.8125 & 175.3125 & 15.9375 & 143.4375 \\ 239.0625 & 111.5625 & 207.1875 & 79.6875 \end{pmatrix}$

۲ - ۵: اجرا الگوریتم **dithering Ordered** با پنجره لغزان تشکیل شده در مرحله قبل.

میدانیم که شبه کد این الگوریتم به صورت زیر است:

**Algorithm 3.1      Ordered Dither**

```

begin
    for  $x = 0$  to  $x_{max}$            // columns
        for  $y = 0$  to  $y_{max}$        // rows
             $i = x \bmod n$ 
             $j = y \bmod n$ 
            //  $I(x, y)$  is the input,  $O(x, y)$  is the output,  $D$  is the dither matrix.
            if  $I(x, y) > D(i, j)$ 
                 $O(x, y) = 1$ ;
            else
                 $O(x, y) = 0$ ;
        end
    end

```

این شبه کد را به صورت زیر پیاده‌سازی میکنیم.



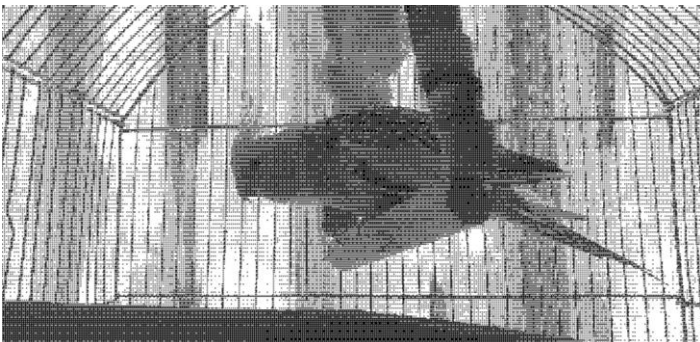
```

for x in range(image_size[0]):
    for y in range(image_size[1]):
        i = x % matrix.shape[0]
        j = y % matrix.shape[1]
        if grayscale_matrix[x][y] > matrix[i][j]:
            dithered_matrix[x][y] = 255
        else:
            dithered_matrix[x][y] = 0

```

برای اندازه پنجره‌های گوناگون، کیفیت تصاویر خروجی متفاوت خواهد بود. چند نمونه از این خروجی‌ها در زیر آورده شده اند.



اندازه پنجره	تصویر خروجی
$w = 2$	
$w = 4$	
$w = 8$	
$w = 16$	