نام و نام خانوادگی: حسنا اویارحسینی	گزارش تمرین دوم درس سیستم های چندرسانه ای
شماره دانشجویی: ۹۸۲۳۰۱۰	تاریخ: اردیبهشت - ۱۴۰۲

" تمرین دوم - بخش عملی – سوالات تشریحی "

سوال ۱)

- با افزایش اندازه ماتریس دیترینگ:

وضوح فضایی الگوی پراکندگی افزایش می‌یابد: به این معنی که جزئیات کوچک‌تر در تصویر را می‌توان با دقت بیشتری حفظ کرد و در نتیجه خروجی با کیفیت بالاتری به دست می‌آید.

بزرگی نویز پراکنده کاهش می‌یابد: به این معنی که مقدار کلی نویز اضافه شده به تصویر کاهش می‌یابد و در نتیجه ظاهری نرم‌تر می‌شود.

تعداد سطوح خاکستری موجود افزایش می‌یابد: این بدان معناست که تصویر خروجی می‌تواند محدوده بیشتری از مقادیر خاکستری داشته باشد، که می‌تواند محدوده تونال و کنتراست کلی را بهبود بخشد.

و با کاهش اندازه این ماتریس تاثیر های برعکسی را خواهیم داشت.

با این حال، افزایش اندازه ماتریس dithering نیز دارای معایبی است:

هزینه محاسباتی اعمال دیترینگ افزایش می‌یابد: این بدان معناست که ماتریس‌های دیترینگ بزرگ‌تر ممکن است آهسته‌تر اعمال شوند، به خصوص برای تصاویر بزرگ‌تر یا برنامه‌های بلادرنگ.

الگوی پراکندگی می تواند بیشتر قابل مشاهده باشد: به این معنی که تصویر خروجی ممکن است شروع به نشان دادن یک الگو یا بافت قابل توجه کند، به خصوص اگر در مقیاس بزرگ مشاهده شود یا با وضوح بالا چاپ شود.

- بله، ماتریس دیترینگ ۳x۳ را می توان برای دیترینگ ترتیبی استفاده کرد. با این حال، در مقایسه با ماتریس های پراکنده بزرگتر، محدودیت هایی دارد. یک ماتریس پراکنده ۳x۳ تنها ۹ مقدار دارد، به این معنی که تنها می تواند ۱۰ سطح خاکستری (شامل سیاه خالص و سفید خالص) تولید کند. این می تواند در نواحی با تغییرات تونال ظریف، اثرات باندینگ شدن قابل مشاهده ای داشته باشد.

این ماتریس با همان فرمول سوال بعد به صئرت زیر بدست می آید:

۳ ۷ ۰

۲ ۵ ۶

۸ ۱ ۴

سوال ۲)

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

```python
def dither_matrix(m:int):
    if m == 1:
        return np.array([[0]])
    else:
        first = (m ** 2) * dither_matrix(int(m/2))
```

```
        second = (m ** 2) * dither_matrix(int(m/2)) + 2
        third = (m ** 2) * dither_matrix(int(m/2)) + 3
        fourth = (m ** 2) * dither_matrix(int(m/2)) + 1
        first_row = np.concatenate((first, second), axis=1)
        second_row = np.concatenate((third, fourth), axis=1)
        return (1/n**2) * np.concatenate((first_row, second_row), axis=0)
```

سوال ۳)

علت آن است که این ماتریس سر ریز خطا بر روی پیکسل های اطراف را به ما نشان میدهد و با توجه به جهت لغزش ماتریس دترینگ بر روی تصویر که از بالا به پایین و از چپ به راست می باشد ما در هر پیکسل که باشیم مقادیر پیکسل هایی که قبل از آن تعیین کرده ایم ثابت شده اند و قابل تغییر نیستند به همین علت مقادیر آن ها یعنی ۵ پیکسل بالا و چپ و خود پیکسل فعلی صفر قرار داده شده اند.

سوال ۴)

این تابع مقدار مربوط به پیکسل فعلی را میگیرد و نزدیک ترین رنگ در مقادیر رنگ کوانتیزه شده را به عنوان خروجی برمیگرداند. یعنی نزدیکترین مقدار کوانتایز شده به نسب رنگ اصلی پیکسل. در پیاده سازی ما با تعیین کردن تعداد کانال هایی که در هر یک از رنگ های RGB میخواهیم داشته باشیم رنگ ها را کوانتایز میکنیم و نزدیک ترین مقدار را به عنوان خروجی برمیگردانیم.