

ملیکا محمدی فخار - ۹۹۵۲۲۰۸۶
تمرین چهارم

.۱

فانکشن‌ها بر اساس فرمول تبدیلات داخل اسلایدها پیاده‌سازی شده‌اند. برای توضیح هر بخش کامنت مناسب داخل کد درج شده است.

```
1 rgb_image = cv2.imread("./content/1.jpg")
2 rgb_image = cv2.cvtColor(rgb_image, cv2.COLOR_BGR2RGB)
3 rgb_image
```

ndarray (427, 640, 3) show data



```
def RGB_to_CMYK(r, g, b, RGB_SCALE=255, CMYK_SCALE=100):
    """
    Convert RGB color values to CMYK color values.

    Args:
        r (int): Red component value (0 to 255).
        g (int): Green component value (0 to 255).
        b (int): Blue component value (0 to 255).
        RGB_SCALE (int, optional): Scale factor for RGB values. Defaults to 255.
        CMYK_SCALE (int, optional): Scale factor for CMYK values. Defaults to 100.

    Returns:
        tuple: CMYK color values as floats in the range of 0 to CMYK_SCALE.
    """
    # Convert RGB [0, 255] to [0, 1]
    r_norm = r / RGB_SCALE
    g_norm = g / RGB_SCALE
    b_norm = b / RGB_SCALE

    # Find the maximum of normalized RGB values
    max_rgb = max(r_norm, g_norm, b_norm)

    # Calculate key (black) component
    k = 1 - max_rgb

    # Avoid division by zero if black is 1
    if k == 1:
        return 0, 0, 0, CMYK_SCALE

    # Calculate CMY values
    c = (1 - r_norm - k) / (1 - k)
    m = (1 - g_norm - k) / (1 - k)
    y = (1 - b_norm - k) / (1 - k)

    # Scale CMYK values to the desired scale
    c = c * CMYK_SCALE
    m = m * CMYK_SCALE
    y = y * CMYK_SCALE
    k = k * CMYK_SCALE

    return c, m, y, k
```

```

def CMYK_to_RGB(c, m, y, k, RGB_SCALE=255, CMYK_SCALE=100):
    """
    Convert CMYK color values to RGB color values.

    Args:
        c (float): Cyan component value (0 to CMYK_SCALE).
        m (float): Magenta component value (0 to CMYK_SCALE).
        y (float): Yellow component value (0 to CMYK_SCALE).
        k (float): Key (black) component value (0 to CMYK_SCALE).
        RGB_SCALE (int, optional): Scale factor for RGB values. Defaults to 255.
        CMYK_SCALE (int, optional): Scale factor for CMYK values. Defaults to 100.

    Returns:
        tuple: RGB color values as integers in the range of 0 to RGB_SCALE.
    """
    # Convert CMYK [0, CMYK_SCALE] to [0, 1]
    c_norm = c / CMYK_SCALE
    m_norm = m / CMYK_SCALE
    y_norm = y / CMYK_SCALE
    k_norm = k / CMYK_SCALE

    # Calculate RGB values
    r = RGB_SCALE * (1 - min(1, c_norm * (1 - k_norm) + k_norm))
    g = RGB_SCALE * (1 - min(1, m_norm * (1 - k_norm) + k_norm))
    b = RGB_SCALE * (1 - min(1, y_norm * (1 - k_norm) + k_norm))

    return int(r), int(g), int(b)

```

```

import math

def RGB_to_HSI(r, g, b):
    """
    Convert RGB color values to HSI color space.

    Args:
        r (int): Red component value (0 to 255).
        g (int): Green component value (0 to 255).
        b (int): Blue component value (0 to 255).

    Returns:
        tuple: HSI color values (hue, saturation, intensity).
    """
    # Convert RGB values to the range [0, 1]
    r_norm = r / 255.0
    g_norm = g / 255.0
    b_norm = b / 255.0

    # Calculate intensity
    i = (r_norm + g_norm + b_norm) / 3.0

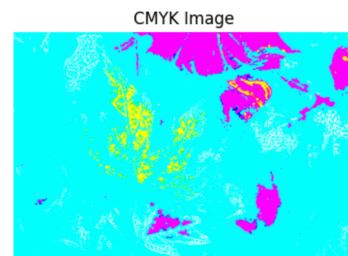
    # Calculate saturation
    if i > 0:
        s = 1 - min(r_norm, g_norm, b_norm) / i
    else:
        s = 0

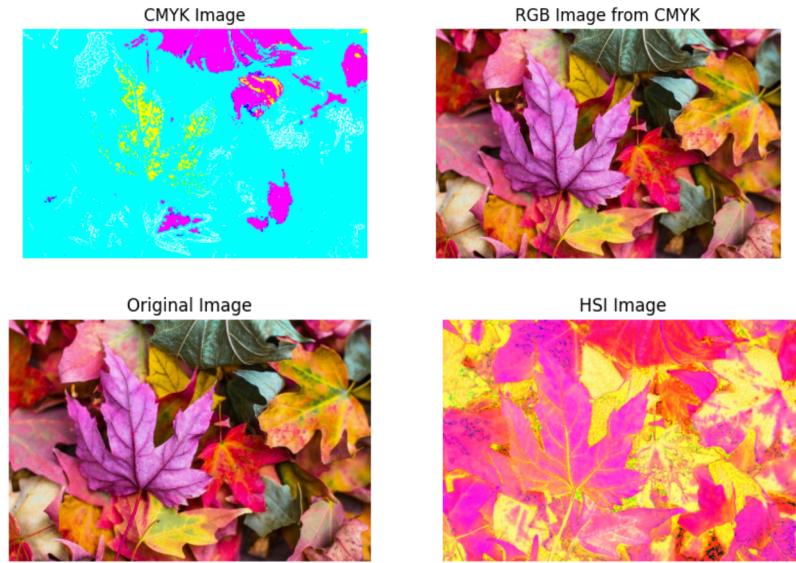
    # Avoid division by zero for hue calculation
    if r_norm == g_norm == b_norm:
        h = 0
    else:
        theta = math.acos(0.5 * ((r_norm - g_norm) + (r_norm - b_norm)) / math.sqrt((r_norm - g_norm) ** 2 + (r_norm - b_norm) * (g_norm - b_norm)))
        if b_norm <= g_norm:
            h = math.degrees(theta)
        else:
            h = 360 - math.degrees(theta)

    return h, s, i

```

خروجی فانکشن‌های بالا (به ترتیب):





۲. همانطور که به وسیله کامنت‌ها توضیح داده شده، ابتدا سایز دو تصویر در صورت تقاؤت، مشترک شده است. سپس اختلاف آن‌ها محاسبه شده است.

```
def diff (image1, image2):
    """
    Compute the absolute difference between two images, with unchanged pixels set to black.

    Args:
        image1 (numpy.ndarray): First input image.
        image2 (numpy.ndarray): Second input image.

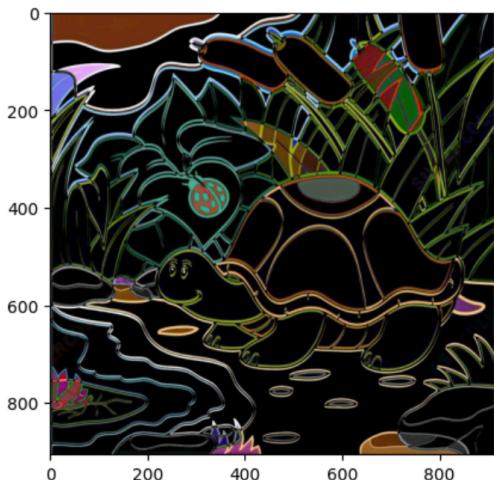
    Returns:
        numpy.ndarray: Image showing the absolute difference between the two input images.
    """
    # Resize image1 to match the image2
    resized_image1 = cv2.resize(image1, (image2.shape[1], image2.shape[0]))

    # Compute difference between images
    diff_image = cv2.absdiff(resized_image1, image2)

    return diff_image
```

نمایش خروجی:

همانطور که مشخص است. پیکسل‌هایی که در آن دو تصویر اختلاف داشته‌اند به صورت رنگی (اختلاف در کانال‌های رنگی) و پیکسل‌هایی که به یک رنگ بوده‌اند به صورت سیاه نمایش داده شده‌اند.



.۳
الف.

$$H = \sum_{(x,y) \in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

برای محاسبه مشتق در پنجره تعیین شده، جذر مجموع مربع درایه‌ها را محاسبه می‌کنیم:

$$H = \begin{bmatrix} 56 & 56 \\ 56 & 60 \end{bmatrix}$$

.ب.

$$\det = 56 * 60 - 56 * 56 = 224$$

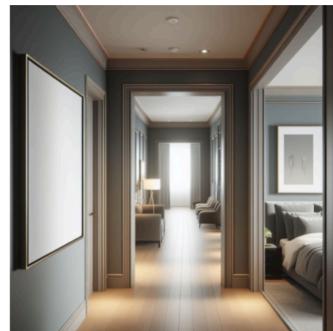
$$\text{trace} = 56 + 60 = 116$$

$$R = 224 - 0.04 * 116 * 116 = -314.24$$

.ج.

از آنجا که مقدار R منفی است (یکی از مقادیر ویژه بزرگ و دیگری کوچک)، لبه داریم.

فرایند خواندن و نمایش دادن تصاویر به منوال سوال قبل پیش می‌رود:



در مرحله بعد روبان مشکی به تصویر اضافه می‌شود. توضیح کد به صورت مرحله به مرحله:

۱. ایجاد ماسک روبان:

یک ماسک سیاه با همان ابعاد تصویر grandpa ایجاد می‌شود.

۲. تعریف نقاط روبان:

نقاط روبان به عنوان مختصات در فضای تصویر تعریف می‌شوند.

۳. پر کردن محدوده روبان:

محدوده‌ای که توسط نقاط روبان محصور شده است با رنگ سفید (255) در ماسک پر می‌شود.

۴. کپی کردن تصویر اصلی:

یک کپی از تصویر اصلی (grandpa) برای کار با آن ایجاد می‌شود.

۵. تبدیل منطقه روبان به رنگ سیاه:

منطقه متناظر با منطقه سفید در ماسک روبان در کپی تصویر اصلی به رنگ سیاه $([0, 0], [0, 0])$ تنظیم می‌شود.

۶. نمایش تصویر تغییر یافته:

تصویر تغییر یافته با روبان سیاه با استفاده از تکنیکهای ماسک‌گذاری و رسم نمودار با استفاده از Matplotlib پلات می‌شود.

۷. نمایش تصویر:

تصویر تغییر یافته با روبان سیاه بدون محور نمایش داده می‌شود.



هدف بخش بعدی که اضافه کردن یک جعبه سیاه به تصویر اتاق است.

۱. ایجاد ماسک سیاه:

یک ماسک سیاه با همان ابعاد تصویر اتاق ایجاد می‌شود.

۲. پر کردن محدوده جعبه:

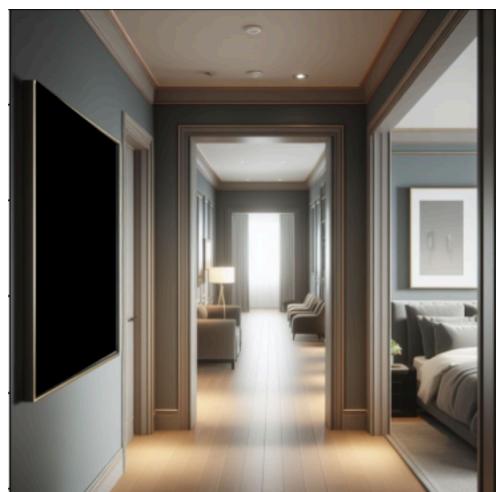
محدوده‌ای که توسط نقاط جعبه مشخص شده است با رنگ سفید (255) در ماسک پر می‌شود.

۳. تبدیل منطقه جعبه به رنگ سیاه:

منطقه متناظر با منطقه سفید در ماسک جعبه در تصویر اتاق به رنگ سیاه ($[0, 0, 0]$) تنظیم می‌شود.

۴. نمایش تصویر تغییر یافته:

تصویر تغییر یافته با جعبه سیاه با استفاده از تکنیک‌های ماسک‌گذاری و رسم نمودار با استفاده از Matplotlib پلاس می‌شود.



سپس فانکشن project_image را به این صورت تعریف می‌کنیم:

محاسبه نقاط مبدا و مقصد:

۱. در ابتدا، نقاط مبدا (یا نقاطی که می‌خواهیم به تصویر جدید تبدیل شوند) به عنوان گوشاهای تصویر grandpa تعریف می‌شوند. این گوشاهای شامل چهار نقطه‌ی بیرونی تصویر هستند.

۲. نقاط مقصد نیز به عنوان نقاطی که می‌خواهیم تصویر جدید به آنها تبدیل شود (معمولًاً بر روی دیوار یا سطح دیگری از تصویر اتاق)، مشخص می‌شوند.

۳. محاسبه ماتریس همگرافی:

با استفاده از نقاط مبدا و مقصد، ماتریس همگرافی محاسبه می‌شود. این ماتریس معین می‌کند که چگونه تصویر مبدا (grandpa) به تصویر مقصد (اتاق) تغییر شکل می‌دهد.

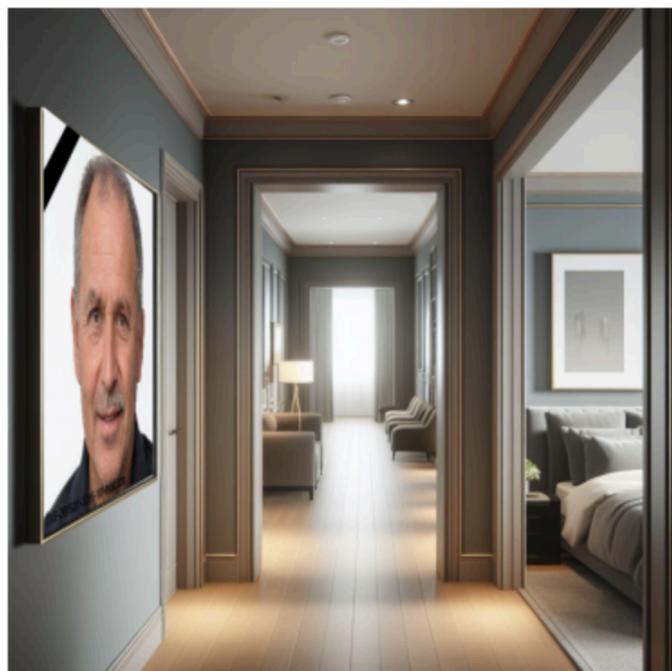
۴. اعمال تبدیل تصویری:

با استفاده از ماتریس همگرافی، تصویر grandpa به اندازه و اندازه‌ی تصویر اتاق تغییر شکل می‌دهد. این عمل توسط تابع cv2.warpPerspective انجام می‌شود.

۵. اضافه کردن تصویر جدید به تصویر اتاق:

تصویر grandpa که به اندازه‌ی تصویر اتاق تغییر شکل یافته است، با تصویر اتاق با استفاده از تابع cv2.addWeighted ترکیب می‌شود. این کار باعث ایجاد ترکیبی از تصاویر می‌شود که در نهایت تصویری با تصویری از اتاق به همراه تصویر grandpa را نشان می‌دهد.

با اعمال این فانکشن بر روی تصاویر داده شده، خروجی به این صورت خواهد بود:



.۵

الف.

از آنجا که درجه آزادی تبدیل affine مقدار ۶ است. با ۳ نقطه می‌توانیم پارامترهای آن را به دست آوریم. ۳ تا از نقاط را در رابطه affine جایگذاری می‌کنیم:

:A نقطه

$$0 * a_{11} + 0 * a_{12} + t_x = 3 \rightarrow t_x = 3$$

$$0 * a_{21} + 0 * a_{22} + t_y = 2 \rightarrow t_y = 2$$

:B نقطه

$$1 * a_{11} + 0 * a_{12} + t_x = 4 \rightarrow a_{11} + t_x = 4$$

$$1 * a_{21} + 0 * a_{22} + t_y = 1 \rightarrow a_{21} + t_y = 1$$

:D نقطه

$$1 * a_{11} + 2 * a_{12} + t_x = 1$$

$$1 * a_{21} + 2 * a_{22} + t_y = 2$$

از حل معادلات بالا خواهیم داشت:

$$a_{11} = 1, a_{12} = -1.5, a_{21} = -1, a_{22} = 0.5$$

بنابراین:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & -1.5 & 3 \\ -1 & 0.5 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

.ب

$$\begin{bmatrix} x'_e \\ y'_e \end{bmatrix} = \begin{bmatrix} 1 & -1.5 & 3 \\ -1 & 0.5 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1.5 \\ 2.5 \end{bmatrix}$$

$$\begin{bmatrix} x'_c \\ y'_c \end{bmatrix} = \begin{bmatrix} 1 & -1.5 & 3 \\ -1 & 0.5 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 3.5 \\ 0.5 \end{bmatrix}$$

تصویری	affine	شباخت	rigid	انتقال	نوع تبدیل
					فاصله جفت نقطات ثابت میماند
					زاویه بین خط ثابت میماند
					خط ها، خط باقی مانند
					زاویه بین هر خط و محور ایکس ثابت میماند
					چهار ضلعی ها، چهار ضلعی باقی می مانند
					خطوط موازی، موازی باقی می مانند
					دایره ها، دایره باقی می مانند
					نسبت بین مساحت دو شکل ثابت باقی می ماند

.الف.

از آنجا که آخرین المنشی X یک است حاصل تقسیم هر المنشی در ۱ خودش می‌شود $(0, 2, 2)$:

$$X = \frac{0}{1} = 0, \quad Y = \frac{2}{1} = 2, \quad Z = \frac{2}{1} = 2$$

حاصل ضرب نقطه سه بعدی X در ماتریس دوربین:

$$P \cdot X = \begin{bmatrix} 5 & -14 & 2 & 17 \\ -10 & -5 & -10 & 50 \\ 10 & 2 & -11 & 19 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

تبديل به کارتزین (در ۱ ضرب می‌شود):

$$\begin{bmatrix} -7 \\ 20 \\ 1 \end{bmatrix}$$

.ب.

$$k = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$f = \frac{5}{0.02} = 250 = f_x = f_y$$

$$c_x = c_y = 500$$

$$\rightarrow k = \begin{bmatrix} 250 & 0 & 500 \\ 0 & 250 & 500 \\ 0 & 0 & 1 \end{bmatrix}$$

ماتریس چرخش:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ماتریس انتقال:

$$T = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

تبديل تركيب انتقال و چرخش:

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

بخش آخر:

$$P'_i = \begin{bmatrix} 250 & 0 & 500 \\ 0 & 250 & 500 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 100 \\ 150 \\ 800 \\ 1 \end{bmatrix} = [425000 \quad 427500 \quad 800]$$

$$P_i = \left[\frac{P'_i[0]}{P'_i[2]}, \frac{P'_i[1]}{P'_i[2]} \right] = [531.25, 546.875]$$