

دانشگاه علم و صنعت

تمرین چهارم مبانی بینایی کامپیووتر

نام و نام خانوادگی:

فرناز خوش دوست آزاد

شماره دانشجویی:

99521253

نام استاد:

دکتر محمدرضا محمدی

(1)

به نوبه که ۱-۲ مراجعه کرده و پس از پیاده سازی توابع خواسته شده، تبدیلات زیر را بر روی تصویر ۱ اعمال کنید(۲۰ نمره)

الف) تبدیل RGB به CMYK و بالعکس (توجه نمایید که مقیاس ۲۵۵ ، RGB و CMYK در صد است.)(۱۰ نمره)

پاسخ:

در ابتدا باید تصویر داده شده را بخوانیم که با استفاده از کتابخانه plt این کار را انجام داده ام.

```
1 # convert BGR to RGB
2 image_bgr = cv2.imread('1.jpg')
3 image_rgb = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2RGB)
4 plt.imshow(image_rgb)
5
<matplotlib.image.AxesImage at 0x7de12445ac0>

```

سپس برای تبدیل های خواسته شده بر روی هر پیکسل از توابع زیر استفاده کرده ام.

```
1 def RGB_to_CMYK(r, g, b, RGB_SCALE = 255, CMYK_SCALE = 100):
2     #####
3     # Your code #
4     R = r / 255
5     G = g / 255
6     B = b / 255
7
8     k = 1 - max(R, G, B)
9     if k == 1:
10        c = m = y = 0
11    else:
12        c = ((1 - R - k) / (1 - k)) * 100
13        m = ((1 - G - k) / (1 - k)) * 100
14        y = ((1 - B - k) / (1 - k)) * 100
15
16    #####
17    k = k * 100
18    return c, m, y, k
```

در این تابع، مقادیر RGB به ترتیب برای قرمز، سبز (G) و آبی (B) به عنوان ورودی گرفته می‌شوند. مراحل انجام شده در این تابع عبارتند از: مقادیر RGB از مقیاس 0 تا 255 به مقیاس 0 تا 1 تبدیل می‌کنیم و سپس مقدار مشکی (k) بر اساس بزرگترین مقدار از G، R و B را محاسبه می‌کنیم. اگر مقدار k برابر 1 باشد، مقادیر cyan، قرمز (m) و زرد (y) را صفر قرار داده و در غیر این صورت، مقادیر m، c و y را محاسبه می‌کنیم.

:CMYK_to_RGB

```

1 def CMYK_to_RGB(c, m, y, k, RGB_SCALE = 255, CMYK_SCALE = 100):
2     #####
3     # Your code #
4     C = c / CMYK_SCALE
5     M = m / CMYK_SCALE
6     Y = y / CMYK_SCALE
7     K = k / CMYK_SCALE
8     r = RGB_SCALE * (1 - C) * (1 - K)
9     g = RGB_SCALE * (1 - M) * (1 - K)
10    b = RGB_SCALE * (1 - Y) * (1 - K)
11    r = round(min(max(r, 0), RGB_SCALE))
12    g = round(min(max(g, 0), RGB_SCALE))
13    b = round(min(max(b, 0), RGB_SCALE))
14    #####
15
16    return r, g, b

```

و در قسمت بعدی تبدیل CMYK_to_RGB را داریم که مانند تصویر بالا از سه تابع برای تبدیلات معکوس یا مشابه با کد بالا استفاده شده و در آخر ذخیره شده اند. مقدار k را نیز به مقیاس 0 تا 100 تبدیل می‌کنیم. و مقادیر y، c، m و k به عنوان خروجی تابع برای استفاده در فضای رنگ CMYK به ازای هر پیکسل را بر می‌گردانیم.

این تابع امکان تبدیل رنگ‌های CMYK (قرمز، CYAN، زرد و مشکی) به فضای رنگ RGB (قرمز، سبز، آبی) را برای هر پیکسل می‌دهد. در این تابع، مقادیر CMYK به ترتیب برای CYAN، قرمز (M)، زرد (Y) و مشکی (K) به عنوان ورودی گرفته می‌گیریم.

مراحل انجام شده در این تابع عبارتند از:

1. تبدیل مقادیر CMYK از مقیاس 0 تا 100 به مقیاس 0 تا 1.

2. محاسبه مقادیر قرمز (Rd)، سبز (g) و آبی (b) با استفاده از فرمول‌های تبدیل CMYK به RGB.

3. چک کردن مقادیر g، r و b برای اطمینان از اینکه در محدوده صحیح رنگ‌ها باقی می‌مانند.

4. گرد کردن مقادیر g، r و b به نزدیکترین عدد صحیح.

5. برگرداندن مقادیر g، r و b به عنوان خروجیتابع برای استفاده در فضای رنگ RGB.

ب) تبدیل RGB به HSI (ده نمره)

پاسخ:

```
1  def RGB_to_HSI(r, g, b):
2      r /= 255.0
3      g /= 255.0
4      b /= 255.0
5      i = (r + g + b) / 3
6      min_val = min(r, g, b)
7
8      if i == 0:
9          s = 0
10     else:
11         s = 1 - (min_val / i)
12     if r == g == b:
13         h = 0
14     else:
15         numerator = 0.5 * ((r - g) + (r - b))
16         denominator = np.sqrt((r - g)**2 + (r - b)*(g - b))
17         theta = np.arccos(np.clip(numerator / denominator, -1, 1))
18         if b <= g:
19             h = theta
20         else:
21             h = 2 * math.pi - theta
22         # h = h * (360 / (2 * math.pi))
23     h = np.degrees(h)
24
25     return h, s, i
```

در این تابع، تبدیل رنگ از فضای رنگ RGB به فضای رنگ HSI (هیو، اشباع، شدت) انجام می‌شود. مراحل انجام شده در این تابع به صورت زیر است:

مقادیر RGB ورودی از مقیاس 0 تا 255 به مقیاس 0 تا 1 نرمال شده و به ترتیب به عنوان g، r و b در نظر گرفته می‌شوند. سپس مقدار شدت (i) بر اساس میانگین مقادیر رنگی g، r و b حساب می‌کنیم. برای محاسبه اشباع (s) بر اساس مقدار کمینه از g، r و b نیز از فرمول بالا استفاده می‌کنیم و اگر مقدار شدت صفر باشد، اشباع نیز صفر قرار داده می‌شود. محاسبه هیو (h) بر اساس مقادیر رنگی g، r و b. در صورتی که رنگ یک

سایه از خاکستری باشد، هیو تعریف نشده است اما به طور معمول صفر قرار داده می شود و سپس زاویه hue را بر اساس مقادیر g ، r و b به دست آورده و آن را تبدیل به درجه می کنیم و در آخر مقادیر را برمی گردانیم. این تابع به ما امکان می دهد تا از مدل رنگی HSI برای توصیف رنگ ها استفاده کنیم.

در نوتبوک قسمتی وجود دارد که باید همه ی پیکسل های تصویر خود را تبدیل به فرمت خواسته شده کرده و سپس تصویر خود را برگردانیم که برای هر کدام از آنها از دو حلقه ی تو در تو استفاده کرده ام که در x و y های پیمایش کرده و مقادیر مناسب را جایگزین می کند و سپس از cv2 برای نشان دادن نتیجه ی آخر استفاده کرده ام که در زیر نتایج آخرب نشان داده شده اند.

RGB_TO_CMYK:

```

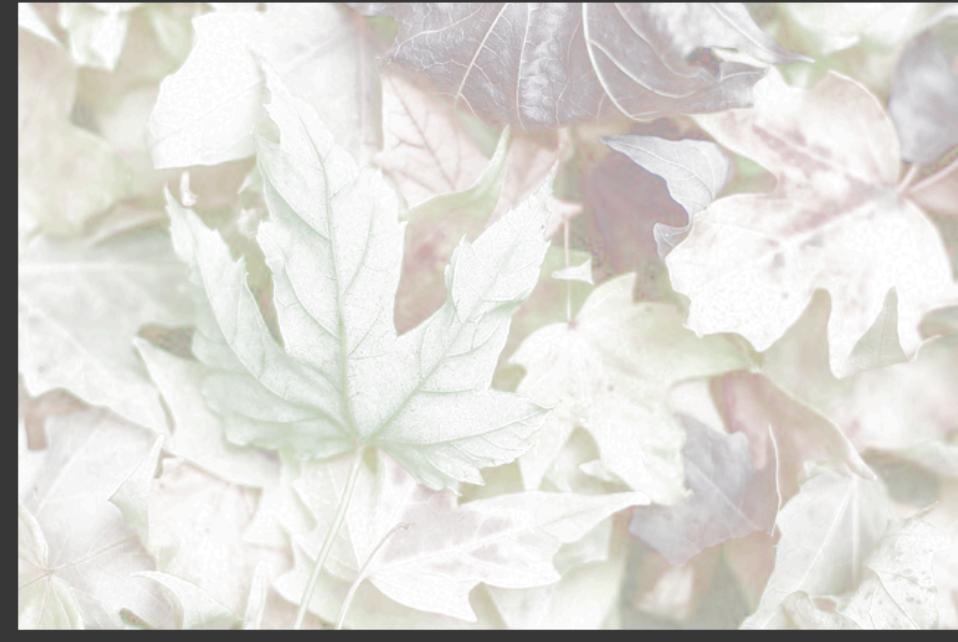
1  # utilize RGB_TO_CMYK for every pixel of picture.
2  def apply_rgb_to_cmyk(image):
3      height, width, _ = image.shape
4      CMYK = np.zeros((height, width, 4), dtype=np.float32)
5      for i in range(height):
6          for j in range(width):
7              r, g, b = image[i, j]
8              CMYK[i, j] = RGB_to_CMYK(r, g, b)
9      return CMYK
10 image_path = '1.jpg'
11 image = Image.open(image_path)
12 image_np = np.array(image)
13 cmyk_image = apply_rgb_to_cmyk(image_np)
14 plt.imshow(cmyk_image)

WARNING:matplotlib.image:Clipping input data to the valid range for imshow<matplotlib.image.AxesImage at 0x7de1243b8730>


```

می بینیم که با استفاده از plt نتیجه به صورت بالا است، حال می خواهیم نتیجه را با استفاده از cv2 ببینیم:

```
1 image_path = '1.jpg'
2 image = Image.open(image_path)
3 image_np = np.array(image)
4 cmyk_image = apply_rgb_to_cmyk(image_np)
5 cv2.imshow(cmyk_image)
6 cv2.waitKey(0)
7 cv2.destroyAllWindows()
8
```



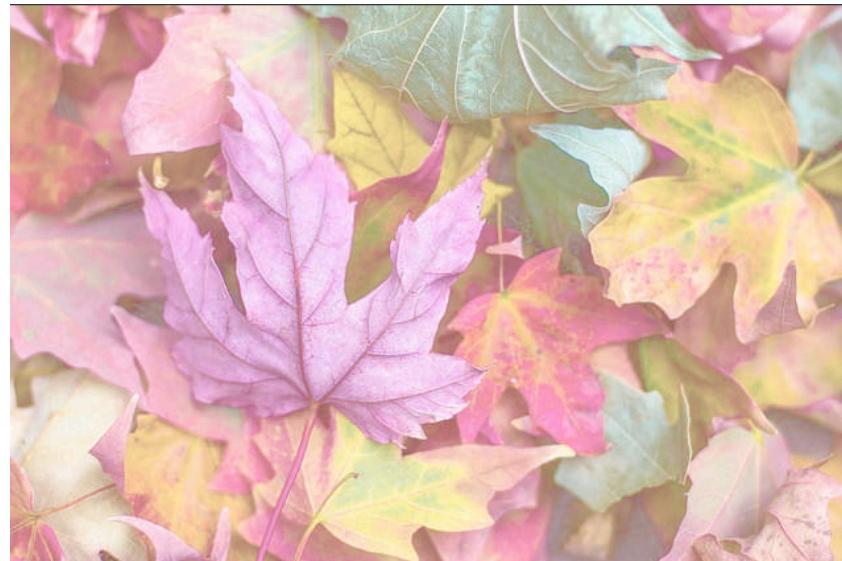
که می بینیم با توجه به تقاضات کتابخانه ها نتیجه مقاوتمانند است.

حال به سراغ کتابخانه دیگری برای CMYK می رویم و می بینیم که نتیجه ای آن نزدیک به جواب ماست که در فایل مربوطه ذخیره شده و کد آن نیز به صورت زیر است:

```
1 # to show the cmyk image
2 def create_cmyk_image(cmyk_np):
3     cmyk_uint8 = np.clip(cmyk_np, 0, 100).astype('uint8')
4     cmyk_image = Image.fromarray(cmyk_uint8, 'CMYK')
5     cmyk_image.show()
6     cmyk_image.save('RGB_TO_CMYK.jpg')
7     print("your result saved in RGB_TO_CMYK.jpg")
8
9 create_cmyk_image(cmyk_image)
10
```

your result saved in RGB_TO_CMYK.jpg

و تصویر آن نیز به شکل زیر است:



می بینیم که تصویر بالا با تصویر خروجی مانتها از لحاظ شدت نوری و تفاوت دارد و این عکس پس از ران شدن کد مربوطه در OS ذخیره می شود که در زیپ نیز قرار داده شده است.

حال به سراغ تابع بعدی می رویم.

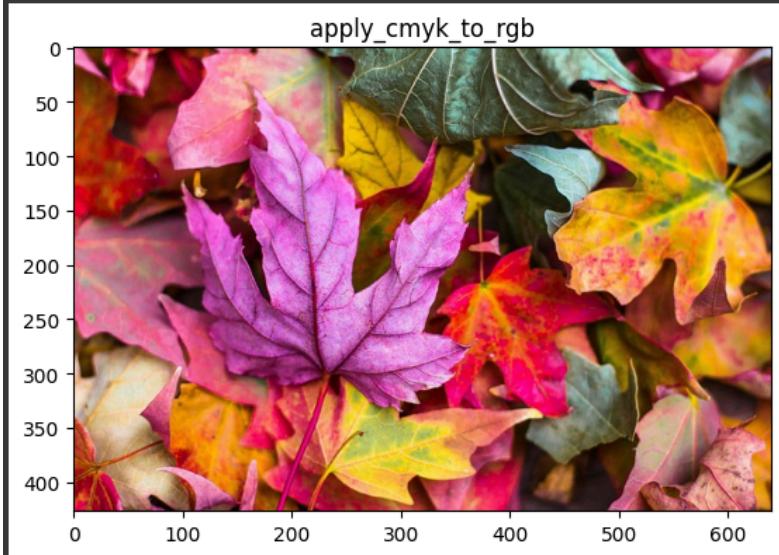
CMYK_TO_RGB:

حال خروجی تابع قبلی را گرفته و در ورودی ایت تابع قرار می دهیم و مانند دفعه‌ی قبل برای همه‌ی پیکسل‌ها با استفاده از دو حلقه‌ی تو در تو برای همه‌ی پیکسل‌ها به کار می گیریم و در آخر جواب را بر می گردانیم.

```

1  def apply_cmyk_to_rgb(image):
2      image_rgb_from_cmyk = np.zeros_like(image_rgb, dtype=np.uint8)
3      for i in range(image.shape[0]):
4          for j in range(image.shape[1]):
5              c, m, y, k = image[i, j]
6              r, g, b = CMYK_to_RGB(c, m, y, k)
7              image_rgb_from_cmyk[i, j] = [r, g, b]
8      return image_rgb_from_cmyk
9  image_rgb = apply_cmyk_to_rgb(cmyk_image)
10 plt.title('apply_cmyk_to_rgb')
11 plt.imshow(image_rgb)
12 plt.show()

```



و در آخر با استفاده از کتابخانه `plt` می بینیم که خروجی ما مانند همان عکس اول است که حاکی از صحت تابع نوشته شده می باشد.

و در آخر تابع تبدیل برای `RGB_TO_HSI` را داریم که تابع آن به صورت زیر است:

```

1  def apply_RGB_to_HSI(image_path):
2      rgb_image = cv2.imread(image_path)
3      # Convert each pixel to HSI
4      hsi_image = np.zeros_like(rgb_image, dtype=np.float32)
5      for row in range(rgb_image.shape[0]):
6          for col in range(rgb_image.shape[1]):
7              r, g, b = rgb_image[row, col]
8              h, s, i = RGB_to_HSI(r, g, b)
9              hsi_image[row, col] = [h, s, i]
10     hsi_image[..., 0] = hsi_image[..., 0] / 360
11     # Save the HSI image as a JPEG file
12     hsi_image_path = "RGB_to_HSI.jpg"
13     cv2.imwrite(hsi_image_path, hsi_image)
14     plt.imshow(hsi_image, cmap = 'hsv', vmin = 0, vmax = 255)
15     print(f"HSI image saved as {hsi_image_path}")
16     image_path = "1.jpg"
17     apply_RGB_to_HSI(image_path)

WARNING:matplotlib.image:Clipping input data to the valid range for imshow
HSI image saved as RGB_to_HSI.jpg


```

پس از اعمال مراحلی که برای تبدیلات پیشین نیز استفاده کردیم، عکس به دست آمده را در فایل **RGB_TO_HSI.JPG** ذخیره کرده ایم که تصویر خروجی آن صفحه ای سیاه است و در صورتی که تقسیم بر 360 را حذف کنم تصویری خواهیم داشت که ترکیبی از سیاه و آبی کربنی می باشد. اما به دلیل این که plt جزئیات بیشتری را نشان می دهد لذا از این کتابخانه برای خروجی بهتر استفاده کرده ام و به دلیل استفاده از 'cmap = 'hsv' خروجی بهتری را مشاهده می کنیم.

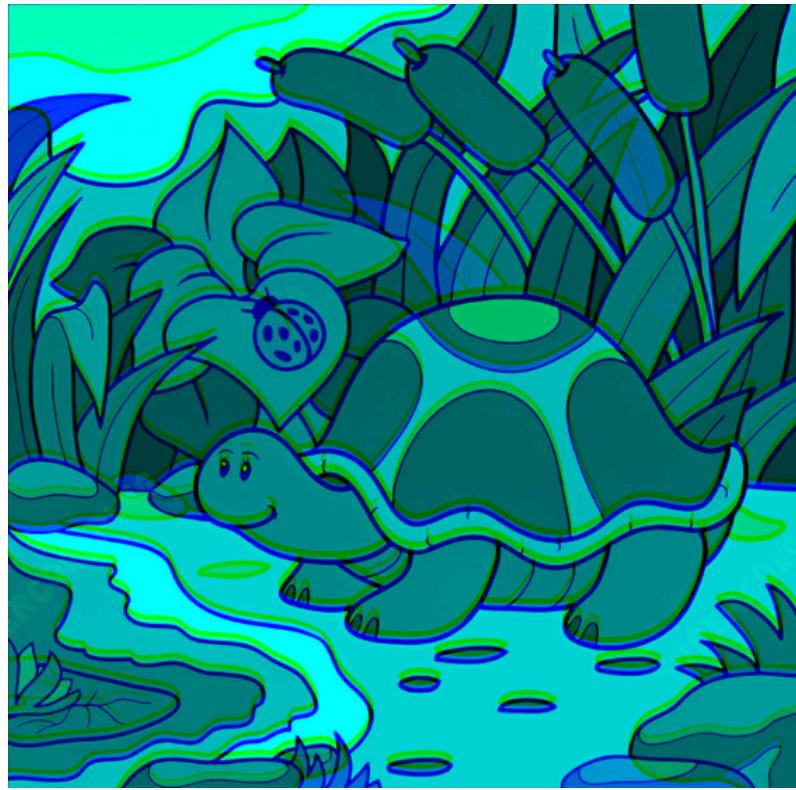
به نوبت 2- q1 مراجعه کرده و با استفاده از روش مشابه با روش گفته شده در کلاس، با طراحی تابعی مقاوتهای موجود در دو عکس jpg.2 و jpg.3 را بیابید(10نمره).

پاسخ:

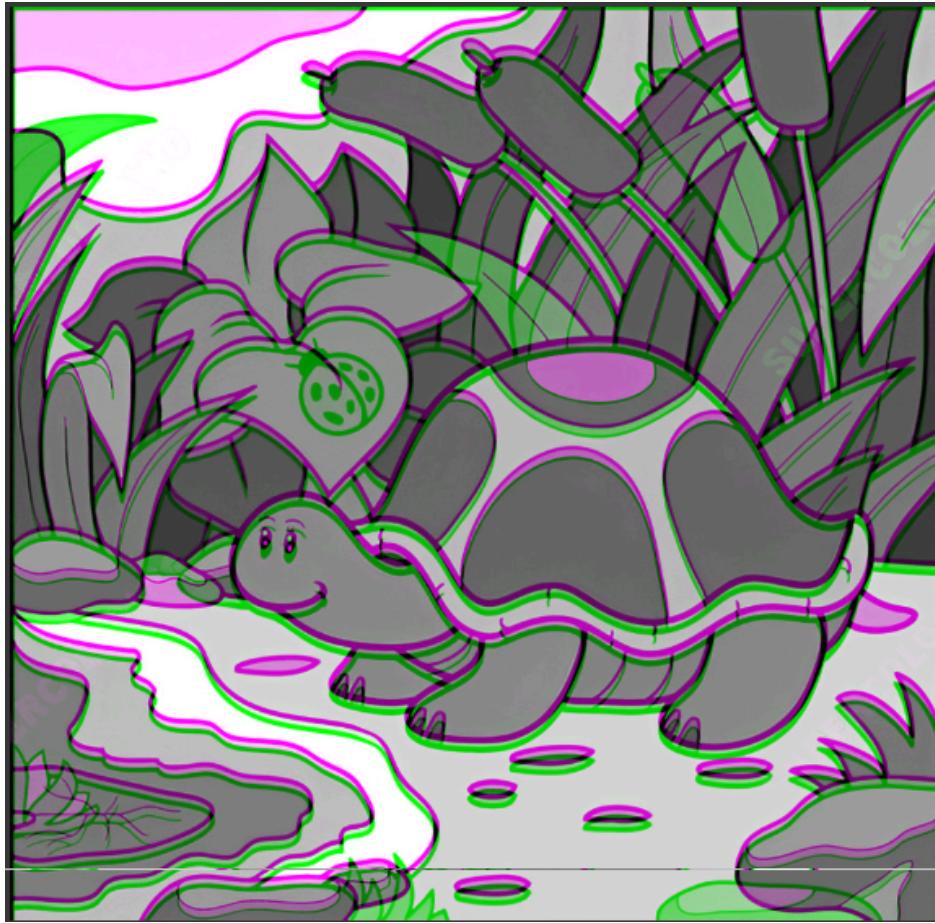
در قسمت دوم همانطور که در کد دیده می شود ابتدا دو عکس را با استفاده از `CV2.imread` می خوانیم و سپس به تعریف تابع `diff` می پردازیم. در ابتدا باید دو عکس خود را به یک سایز تبدیل می کنیم که برای این کار می توانیم از `cv2.resize` استفاده کنیم. کاری که برای `Resize` کردن در این تابع انجام دادم این بود که مینیم طول و عرض دو عکس را حساب کردم و سپس طول و عرض هر دو را برابر با مقدار مینیم قرار دادم و سپس آرایه ای سه بعدی تعریف کردم که طول و عرض آن مانند عکس های `resize` شده بود و بعد آخر آن سه کanal دارد و کanal اول آن را به عکس نسبت دادم و کanal های دیگر را یا برابر با صفر قرار دادم و برابر با عکس دوم گذاشتم و در اینجا به دلیل جایگشت های مقاووت رنگ های به دست آمده می توانند رنگ های مقاووتی به ما تحويل دهند که دو مورد از آنها را در کد خود با نام `diff1` و `diff` قرار دادم و برای ورودی این دو تابع نیز عکس 2 و 3 را بدون کanal خوانده و سپس در ورودی این دو تابع قرار دادم. یکی از توابع نوشته شده در زیر موجود است:

```
1  def diff(image1, image2):
2      h1 = int(image1.shape[1])
3      w1 = int(image1.shape[0])
4      h2 = int(image1.shape[1])
5      w2 = int(image2.shape[0])
6      new_height = min(h1, h2)
7      new_width = min(w1, w2)
8      result = np.zeros((new_width, new_height, 3), np.uint8)
9      result[:, :, 0] = image1[:new_width, :new_height]
10     result[:, :, 1] = image2[:new_width, :new_height]
11     # result[:, :, 2] = image2[:new_width, :new_height]
12     result[:, :, 2] = 0
13
14     return result
15
```

و جوابهای مقاووت آن با کanal های مقاووت به صورت زیر می باشد که هر رنگ در تصویر زیر معنای خاص خود را دارد.



این نتیجه‌ی تابع اول است که مقدار ۲ برای همه‌ی سلول‌ها صفر در نظر گرفته شده است. لذا جاهایی که رنگ فیروزه‌ای دیده می‌شود به معنای این نیست که تغییری رخ نداده است. اما در جاهایی که تصویر آبی است، یعنی آن قسمت در عکس اول بوده اما در عکس دوم وجود نداشته است. حال تابعی دیگر را می‌بینیم که داریم:



در این تصویر که به تابع diff1 نسبت داده می شود، مقدار سبز به تصویر اول و مقدار آبی و قرمز به تصویر دوم نسبت داده شده است و چون ترکیب دو رنگ آبی و قرمز در آخر بنفس دیده می شود، لذا قسمت هایی که در تصویر اول وجود دارد اما در تصویر دوم نیست به رنگ بنفس دیده می شود و جاهایی که در تصویر دوم هست، اما در تصویر اول نیست به رنگ سبز دیده می شود و در آخر جواب به صورت بالا است.

(3)

مشتق افقی و عمودی یک تصویر را حساب کردیم(25 نمره).

$I_x = dI / dx$					$I_y = dI / dy$				
3	2	1	-1	-1	2	3	1	1	-1
4	3	2	0	-1	2	3	2	-1	1
4	3	4	2	1	2	4	4	1	2
1	1	3	2	2	-1	0	3	2	3

a. ماتریس هریس را با پنجره ۳ در ۳ برای پنجره مشخص شده حساب کنید(10 نمره)

پاسخ:

از الگوریتم harris برای تشخیص گوشه های عکس استفاده می کنیم. که در جاهایی که شدت رنگ در جهات مختلف به طور ناگهانی تغییر پیدا می کند، به معنای وجود گوشه ها در عکس است. که در ابتدا باید گرادیان را در جهات مختلف حساب کنیم و سپستابع پاسخ harris را به دست آوریم و نقاطی که از threshold کمتر هستند را حذف می کنیم و در آخر نیز گوشه ها را محلی سازی می کنیم. که فرمول آن نیز در زیر نوشته شده است.

$$E(u, v) \approx \sum_{x,y} w(x, y) [u I_x + v I_y]^2 = \sum_{x,y} w(x, y) (u^2 I_x^2 + 2uv I_x I_y + v^2 I_y^2)$$

$$= [u \quad v] \left(\sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M \triangleq \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

که همانطور که دیده می شود مشتق افقی و عمودی داده شده اند. لذا فقط باید بر اساس فرمول بالا، مقادیر ماتریس harris را حساب کنیم. ابتدا مقدار I_x و I_y را حساب می کنیم که داریم:

$$I_x = 3^{**}2 + 2^{**}2 + 0^{**}2 + 3^{**}2 + 4^{**}2 + 2^{**}2 + 1^{**}2 + 3^{**}2 + 2^{**}2 = 56$$

$$I_y = 3^{**}2 + 2^{**}2 + (-1)^{**}2 + 4^{**}2 + 4^{**}2 + 1^{**}2 + 0^{**}2 + 3^{**}2 + 2^{**}2 = 60$$

$$I_x * I_y = (3 * 3) + (2 * 2) + (0 * -1) + (3 * 4) + (4 * 4) + (2 * 1) + (3 * 3) + (1 * 0)$$

$$+ (3 * 3) + (2 * 2) = 56$$

لذا ماتریس ما به صورت زیر است:

$$\text{Harris_matrix} = \begin{bmatrix} 56 & 56 \\ 56 & 60 \end{bmatrix}$$

B. مقدار زیر را برای پنجره مشخص شده محاسبه کنید(مقدار k را 0.04 در نظر بگیرید)(5 نمره)

$$R = \det(M) - k \operatorname{trace}(M)^2$$

پاسخ:

حال در ابتدا باید مقدار \det و trace ماتریس harris را حساب کنیم:

$$\det(\text{Harris_matrix}) = (56*60) - (56*56) = 224$$

$$\operatorname{trace}(\text{Harris_matrix}) = 56 + 60 = 116$$

لذا مقدار R برابر است با:

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

C. پنجره مشخص شده یک ناحیه ... (لبه/گوشه/خت) می باشد. دلیل خود را توضیح دهید(10 نمره)

پاسخ:

تا جایی که یاد گرفتیم در صورتی که R مقدار بالایی داشته باشد و هم مشتق افقی و هم مشتق عمودی بالا باشند به معنای وجود گوشه است که در اینجا می بینیم مقدار R هم مثبت و هم نسبتاً بزرگ است و $|x|$ و $|y|$ هر کدام بزرگ هستند که به معنای وجود گوشه در این نقطه است. اما اگر فقط یکی از مشتق ها بزرگ بودند حاکی از وجود یک لبه ای افقی یا عمودی بود و در صورتی که R مقدار کوچکی داشت یعنی قسمت flat تصویر را مورد بررسی قرار داده ایم و اگر مقدار منفی بود نیز به معنای لبه بود.

(4)

صادق می خواهد خانه‌ی خود را طراحی کند و از E-DALL برای طراحی اتفاقهای خود کمک گرفته است. اما برای راهروی خانه خود می خواهد عکس پدر بزرگ فوت شده خود را بروی دیوار سمت چپ قرار دهد اما این هوش مصنوعی قابلیت دریافت عکس بصورت ورودی را ندارد؛ پس از شما میخواهد که عکس پدر بزرگ او را بروی قاب عکس سمت چپ راهرو قرار دهید. عکس پدر بزرگ او ربان سیاه ندارد پس ابتدا با استفاده از کد یک ربان سیاه روی عکس ایجاد کنید. مختصات نقاط اطراف قاب را بصورت دستی مشخص کنید. بخش مربوط به ایجاد ربان سیاه بر روی عکس امتیازی است. (خواسته های سوال را در نوت بوک q4 پیاده سازی کنید). (15 نمره + 5 نمره امتیازی)

پاسخ:

در این سوال در ابتدا در تصویر room و grandpa را با استفاده از کتابخانه cv2 نشان می دهیم و در قسمت بعدی باید روبانی مشکی برای پدربزرگ بذاریم که من با استفاده از کد زیر سمت بالای چپ عکس را به پهنای 130 پیکسل به صورت مورب سیاه کردم که به معنای روبان عکس پدربزرگ می باشد.

```
1 distance = 0
2 for i in range(0, 4 * int(grandpa.shape[0] / 7)):
3     count = 0
4     for j in range(3 * int(grandpa.shape[1] / 7), grandpa.shape[1]):
5         if count < 130 and j + distance < grandpa.shape[1]:
6             grandpa[i, j + distance] = [0, 0, 0]
7             count += 1
8     distance += 1
9 cv2.imshow(grandpa)
10 cv2.waitKey(0)
11 cv2.destroyAllWindows()
12 cv2.imwrite('grandpa_with_black_ribbon.jpg', grandpa)
```

و نتیجه آن در فایل نوشته شده ذخیره شد که به صورت زیر می باشد.



this-person-does-not-exist.com

سپس با استفاده از کد زیر نقاط گوشی قاب عکس را تخمین زده ام که برای این کار مختصات های مختلف را امتحان کرده ام و برای هر کدام دایره ای فیروزه ای کشیده ام تا نقاط گوشی ای قاب عکس معلوم شوند.

```

1 import copy
2 center_coordinates = [(60, 170), (225, 285), (60, 805), (225, 720)]
3 radius = 20
4 color = (255, 255, 0)
5 thickness = 2 # Pixel
6 room_with_circle = room.copy()
7 for center in center_coordinates:
8     cv2.circle(room_with_circle, center, radius, color, thickness)
9
10 cv2.imshow(room_with_circle)
11 cv2.waitKey(0)
12 cv2.destroyAllWindows()

```

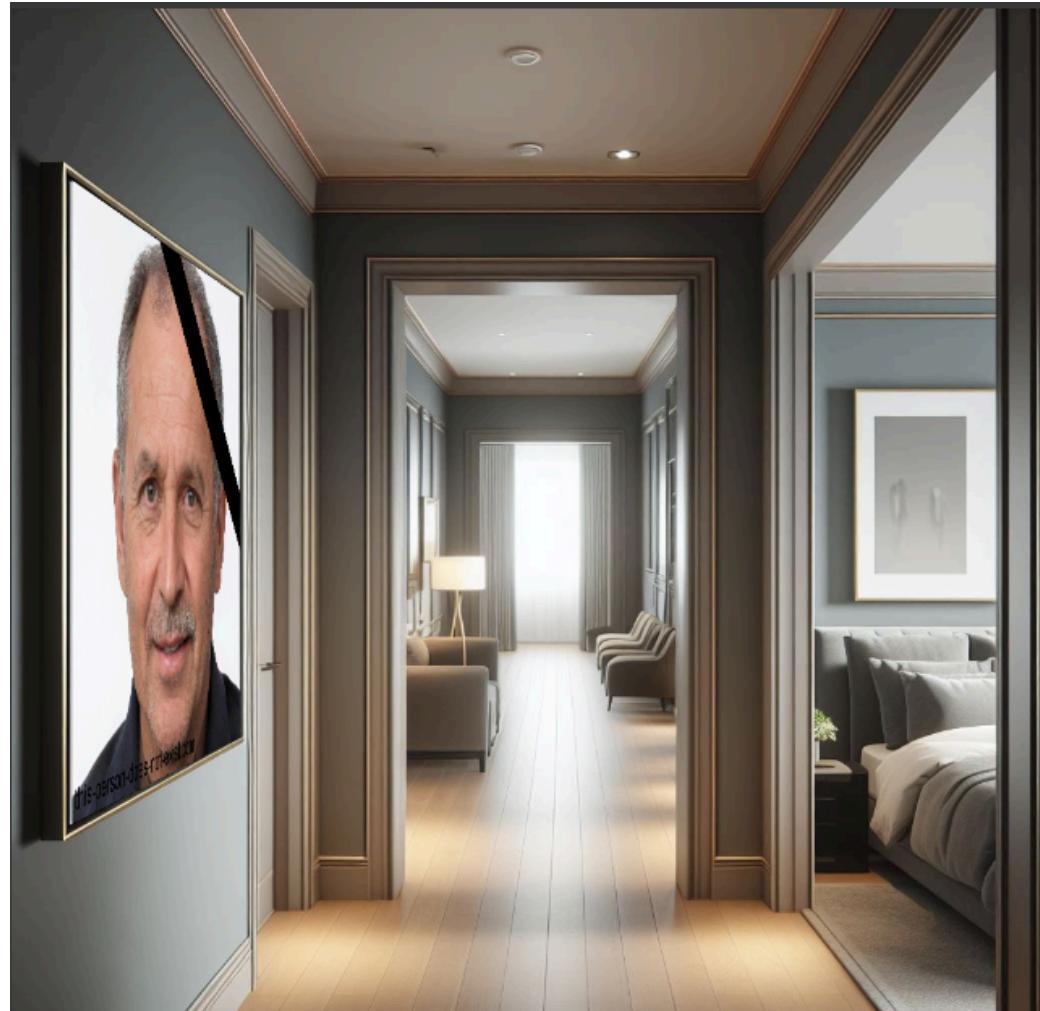
با استفاده از نقاط بالا تا حد خوبی گوشی های عکس تخمین زده شده اند. حال با استفاده از cv2.getPerspectiveTransform(pts_src, pts_dst) و نقاط مبدا و مقصد می توانیم تبدیل یافته عکس پدربرگ را داشته باشیم و سپس با استفاده از یک mask می توانیم عکس پدربرگ را منطبق بر عکس اتاق کنیم که با کد زیر این تبدیلات انجام شد:

```

1 pts_src = np.array([
2     [0, 0],
3     [grandpa.shape[0], 0],
4     [0, grandpa.shape[1] - 1],
5     [grandpa.shape[0] - 1, grandpa.shape[1] - 1],
6 ], dtype='float32')
7
8 pts_dst = np.array([[60, 170], [225, 285], [60, 805], [225, 720]], dtype = 'float32')
9 matrix = cv2.getPerspectiveTransform(pts_src, pts_dst)
10 transformed_image = cv2.warpPerspective(grandpa, matrix, (room.shape[0], room.shape[1]))
11
12 # Create a mask from the warped image
13 mask = cv2.warpPerspective(np.ones_like(grandpa, dtype=np.uint8) * 255, matrix, (room.shape[1], room.shape[0]))
14 # Convert mask to boolean
15 mask_bool = mask.astype(bool)
16 # Blend the warped image into the background
17 result = room.copy()
18 result[mask_bool] = transformed_image[mask_bool]

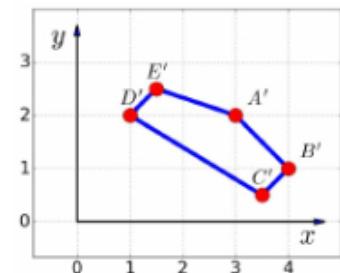
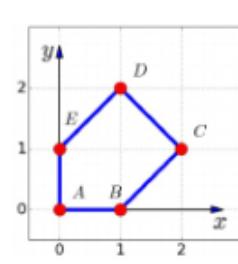
```

که نتیجه آن نیز به صورت زیر است:



(5)

شکل سمت چپ تحت یک تبدیل affine به شکل سمت راست تبدیل شده (10 نمره).



الف) رابطه تبدیل را بدست بیاورید (5 نمره).

ب) مختصات نقاط 'C' و 'E' را بدست بیاورید (5 نمره).

پاسخ:

Subject :

Year. Month. Date. ()

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

مهم 3 خط از این راه حل برای Affine Transformation می باشد

$$A = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad A' = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad B' = \begin{bmatrix} 4 \\ 1 \end{bmatrix}$$

$$O = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad O' = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} x_3 \\ x_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

حال دو نقطه D, B, A بتوانند مطابقت کرد

$$\begin{bmatrix} 4 \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$|3=t_x|, |2=t_y|$$

$$4=a_{11}+t_x \Rightarrow |a_{11}=1|$$

$$1=a_{21}+t_y \Rightarrow |a_{21}=-1|$$

$$1=a_{11}+2a_{12}+t_x \Rightarrow |a_{12}=-3|$$

$$2=a_{21}+2a_{22}+t_y \Rightarrow |a_{22}=\frac{1}{2}|$$

$$A = \begin{bmatrix} 1 & -\frac{3}{2} & 3 \\ -1 & \frac{1}{2} & 2 \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = A \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x'_C \\ y'_C \end{bmatrix} = \begin{bmatrix} 1 & -1,5 & 3 \\ -1 & 0,5 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3,5 \\ 0,5 \end{bmatrix}$$

محصلة C

$$\begin{bmatrix} x'_E \\ y'_E \end{bmatrix} = \begin{bmatrix} 1 & -\frac{3}{2} & 3 \\ -1 & \frac{1}{2} & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1,5 \\ 2,5 \end{bmatrix}$$

محصلة E

(6)

جدول زیر را با مقادیر صحیح/غلط کامل کنید (10 نمره).

تصویری	affine	شماحت	rigid	انتقال	نوع تبدیل
-	-	-	+	+	فاصله جفت نقطات ثابت می ماند
-	-	+	+	+	زاویه بین جفت خط ثابت می ماند
+	+	+	+	+	خط ها، خط باقی مانند
-	-	-	-	+	زاویه ای بین هر خط و محور ایکس ثابت می ماند.
+	+	+	+	+	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.

پاسخ:

(7)

تبدیل هموگرافی(H) را پیدا کنید که هریک دسته خط های

$$\{x = -5 \cdot x + y = 5, 2x + y = 0\} \text{ و } \{x = 5, 2x = y \cdot x + 5 = y\}$$

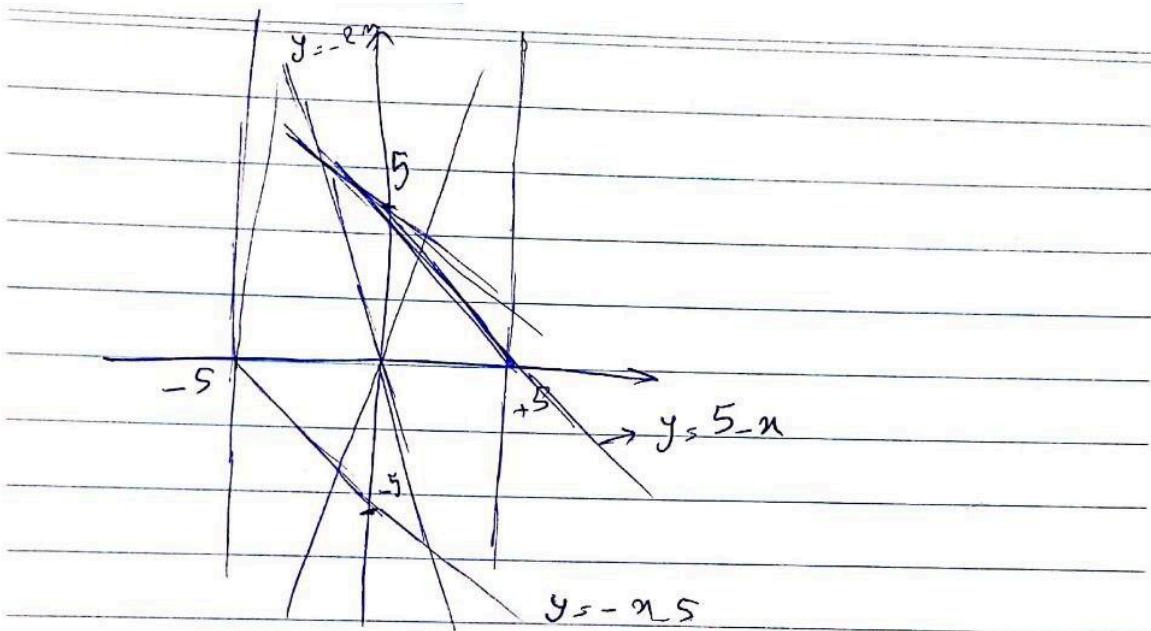
را به مجموعه ای از خطوط موازی تبدیل میکند و دو شرط زیر را ارضاء میکند(10نمره):

$$h33 = 1^\circ$$

نقطه (0,0) را ثابت نگه دارد.

(راهنمایی : تبدیل مورد نظر یک تبدیل 3در3 است که باید هریک از این دسته خطوط را تبدیل به خطوط موازی کند. این کار برای هر یک از دسته ها به صورت جداگانه انجام می شود اما در نهایت باید یک تبدیل واحد داشته باشیم. با توجه به اطلاعاتی که در سوال است مانند موازی بودن خطوط، دو معادله باید تشکیل دهید و مجهول

هایی که میتوان برای آنها مقدار صریح پیدا کرد را پیدا کنید. در نظر داشته باشید که ممکن است نتوانید این ماتریس را به صورت کاملاً صریح پیدا کنید. کافی است در چنین شرایطی شرط های لازم را ذکر کنید.)



Our Homography matrix is

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \xrightarrow{\text{from the constraints}}$$

$$\frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + 1} =$$

bilinear transformation lines

in Matrix is 18 variables and 6 equations
these variables are given by just 3 transformation lines

(8)

الف) با توجه به ماتریس دوربین زیر:

$$P = \begin{bmatrix} 5 & -14 & 2 & 17 \\ -10 & -5 & -10 & 50 \\ 10 & 2 & -11 & 19 \end{bmatrix}$$

که متشکل از ماتریس انتقال و چرخش است و نقطه سه بعدی X در مختصات همگن:

$$X = \begin{bmatrix} 0 \\ 2 \\ 2 \\ 1 \end{bmatrix}$$

7. تبدیل هموگرافی (H) را پیدا کنید که هریک دسته خطهای

$$\{x = -5; x + y = 5, 2x + y = 0\} \text{ و } \{x = 5; 2x = y; x + 5 = y\}$$

را به مجموعهای از خطوط موازی تبدیل میکند و دو شرط زیر را ارضاء میکند (10 نمره):

$$h33 = 1^\circ$$

نقطه (0,0) را ثابت نگه دارد.

(راهنمایی: تبدیل مورد نظر یک تبدیل 3 در 3 است که باید هریک از این دسته خطوط را تبدیل به خطوط موازی

کند. این کار برای هر یک از دسته ها به صورت جداگانه انجام می شود اما در نهایت باید یک تبدیل واحد داشته باشیم.

با توجه به اطلاعاتی که در سوال است مانند موازی بودن خطوط، دو معادله باید تشکیل دهید و مجهول هایی که میتوان برای آنها مقدار صریح پیدا کرد را پیدا کنید. در نظر داشته باشید که ممکن است نتوانید این ماتریس را به صورت کاملاً صریح پیدا کنید. کافی است در چنین شرایطی شرط های لازم را ذکر کنید.)

8. الف) با توجه به ماتریس دوربین زیر:

که متشکل از ماتریس انتقال و چرخش است و نقطه سه بعدی X در مختصات همگن:

• مختصات دکارتی نقطه X در فضای سه بعدی را محاسبه کنید(راهنمایی : از استفاده scaling factor استفاده کنید).

• مختصات دکارتی افکنش X در تصویر را محاسبه کنید(راهنمایی : از ماتریس P استفاده کنید و در نهایت یک نقطه دو بعدی را به دست آورید).

ب) (b) یک دوربین pinhole ایده آل با فاصله کانونی(f) میلیمتر داریم. هر پیکسل 0.02×0.02 میلیمتر است و نقطه اصلی تصویر(cx,cy) در پیکسل (500,500) قرار دارد. مختصات پیکسل از (0,0) در گوش بالا و چپ تصویر شروع میشود.

• ماتریس calibration دوربین(M) برای این پیکربندی دوربین را محاسبه کنید(راهنمایی : ماتریس داخلی دوربین).

• با فرض اینکه چارچوب مختصات جهانی با چارچوب مختصات دوربین هم محور است (به این معنا که مبدأ آنها یکسان است و محورهای آنها هم محور هستند) و مبدأها در pinhole دوربین قرار دارند، ماتریسی (3×4) ای را که نمایانگر تبدیل خارجی بین سیستم مختصات دوربین و سیستم مختصات جهانی است محاسبه کنید(راهنمایی:

این ماتریس مشکل از ماتریس انتقال و ماتریس چرخش است. از صفحه 21 اسالید 1 میتوانید کمک بگیرید).

• با ترکیب نتایجتان از دو قسمت قبل، افکنش نقطه صgne (100,150,1800) را به مختصات تصویر محاسبه کنید

(راهنمایی : باید از اعمال ترکیب ماتریس های محاسبه شده در دو قسمت قبلی برای محاسبه افکنش این نقطه

استفاده کنید)(20 نمره امتیازی).

: پاسخ

404 - error not found