یافتن رنگ های غالب یک تصویر با clustering :

با استفاده از تکنیک های خوشه بندی و الگوریتم k-means می توانیم رنگ های غالب یک تصویر را پیدا کنیم. به کد زیر دقت کنید:

```
n_clusters = 5

#read image
img = cv2.imread("images/felfel-dolme.jpg")

#reshape to a list of pixels
flat_img = img.reshape((-1, 3))

#using k-means to cluster pixels
kmeans = KMeans(n_clusters = n_clusters)
kmeans.fit(flat_img)

#the cluster centers are our dominant colors.
dominant_colors = np.array(kmeans.cluster_centers_, dtype='uint8')

labels = kmeans.labels_

print(dominant_colors)
print(labels)
```

در این کد ابتدا:

```
n_clusters = 5
```

- تعداد خوشه‌ها (گروه‌ها) برای K-Means را مشخص کرده ایم. یعنی می‌خوایم ۵ رنگ غالبِ تصویر رو پیدا کنیم.
- هرچه عدد بزرگ‌تر، رنگ‌های بیشتری به‌عنوان «غالب» برمی‌گردند. (و زمان محاسبه هم بیشتر می‌شود).

```
flat_img = img.reshape((-1, 3))
```

پس از خواند عکس مورد نظر. Shape عکس را تغییر داده ایم. با «بازشکل‌دهی» آرایه، تصویر از شکل سه‌بعدی به ماتریسی دوبعدی تبدیل می‌شود که هر سطر یک پیکسل و هر ستون یکی از سه مؤلفه رنگی است. مقدار -1 به NumPy اجازه می‌دهد اندازه بعدِ سطری را بر اساس تعداد کل پیکسل‌ها محاسبه کند؛ در نتیجه، شکل جدید برابر (height*width, 3)

خواهد بود. این دقیقا چیزی است که k-means به آن احتیاج دارد. هر پیکسل به یک نمونه تبدیل می شود و هر ویژگی یک کانال رنگی.

```
kmeans = KMeans(n_clusters = n_clusters)
kmeans.fit(flat_img)
```

یک شیء از کلاس KMeans متعلق به scikit-learn با تعداد خوشه‌های تعیین‌شده ساخته می‌شود. پارامتر n_clusters بیان می‌کند که الگوریتم باید دقیقاً همین تعداد مرکز خوشه بیابد. سپس الگوریتم K-Means بر مجموعه داده flat_img آموزش داده می‌شود. در طی این فرایند، نقاط (پیکسل‌ها) بر اساس کمینه‌سازی جمع مربعات فاصله تا نزدیک‌ترین مرکز خوشه، دسته‌بندی می‌شوند و مراکز خوشه‌ها به‌صورت تکراری به‌روز می‌گردند تا همگرایی حاصل شود.

```
dominant_colors = np.array(kmeans.cluster_centers_, dtype='uint8')
```

در این مرحله می خواهیم مرکز هر خوشه را پیدا کنیم. ویژگی cluster_centers_ ماتریسی با ابعاد (n_clusters, 3) ارائه می‌کند که هر سطر میانگین رنگ‌های متعلق به یک خوشه است. این مقادیر به‌صورت اعشاری (float)هستند؛ با تبدیل آن به uint8 مقادیر بالا در بازه ۰ تا ۲۵۵ قرار می‌گیرند تا با نمایش یا ذخیره‌سازی رنگ همخوان باشند. بدین‌ترتیب، dominant_colors آرایه‌ای از پنج بردار سه‌کاناله BGR خواهد بود.

```
labels = kmeans.labels_
```

ویژگی labels_ برای هر پیکسل یک برچسب عددی در بازه [0, n_clusters-1] تخصیص می‌دهد که نشان می‌دهد آن پیکسل به کدام خوشه تعلق دارد. طول این بردار برابر با تعداد کل پیکسل‌ها (height*width) است.

محدوده برچسب‌ها در بازه [0, n_clusters - 1] قرار دارد زیرا الگوریتم K-Means برای شناسایی هر خوشه از یک شناسه عددی استفاده می‌کند که از صفر آغاز می‌شود و تا یک واحد کمتر از تعداد خوشه‌ها ادامه می‌یابد. این روش بر اساس شمارش صفرمبنا در زبان‌های برنامه‌نویسی طراحی شده است. بدین ترتیب، وقتی تعداد خوشه‌ها برابر با n_clusters باشد، نخستین خوشه با شناسه ۰ و آخرین خوشه با شناسه n_clusters - 1 مشخص می‌شود. این برچسب‌ها که به‌صورت اعداد صحیح تخصیص می‌یابند، برای هر نمونه (در اینجا هر پیکسل) نشان می‌دهند که آن نمونه به کدام خوشه تعلق دارد و به‌دلیل استفاده از شاخص‌های عددی، ذخیره‌سازی و ارجاع به مراکز خوشه‌ها ساده و کارآمد می‌شود.

```
print(dominant_colors)
print(labels)
```

این دو دستور برای نمایش خروجی‌های اصلی الگوریتم K-Means استفاده می‌شوند. دستور print(dominant_colors) آرایه‌ای با ابعاد (n_clusters, 3) را چاپ می‌کند که هر سطر آن نشان‌دهنده مختصات یک مرکز خوشه در فضای رنگی BGR است و در واقع بیانگر رنگ‌های غالب تصویر پس از خوشه‌بندی

محسوب می‌شود. دستور `print(labels)` یک بردار یک‌بعدی با طول برابر با تعداد کل پیکسل‌های تصویر را چاپ می‌کند که در آن، هر مقدار عددی بیان می‌کند پیکسل متناظر به کدام خوشه تعلق دارد؛ این مقادیر همواره در بازه `[0, n_clusters - 1]` قرار دارند و امکان بازسازی یا تحلیل توزیع خوشه‌ها را فراهم می‌کنند.

```
percentages = np.bincount(labels)/len(flat_img)
percentages
```

این خط کد برای مشخص کردن سهم هر خوشه از کل تصویر استفاده می‌شود. دستور `np.bincount(labels)` تعداد پیکسل‌های هر خوشه را می‌شمارد، یعنی می‌گوید خوشه‌ی صفر چند پیکسل دارد، خوشه‌ی یک چندتا و همین‌طور تا آخر. بعد این تعداد را بر `len(flat_img)` تقسیم می‌کنیم که تعداد کل پیکسل‌های تصویر است. حاصل، نسبتی برای هر خوشه به دست می‌دهد که نشان می‌دهد هر رنگِ غالب چه درصدی از تصویر را پوشانده است. در نهایت، وقتی `percentages`را چاپ کنید، یک آرایه دارید که هر عددش سهم یک رنگ از تصویر را نشان می‌دهد.

```
p_and_c = zip(percentages,dominant_colors)
p_and_c = sorted(p_and_c,reverse=True)
p_and_c
```

این قسمت کد برای ترکیب درصدها با رنگ‌های غالب و مرتب‌سازی آن‌ها استفاده می‌شود. در خط اول، `zip(percentages, dominant_colors)`دو آرایه را به‌صورت جفت‌های (درصد، رنگ) (کنار هم قرار می‌دهد تا هر درصد با رنگ مربوط به همان خوشه جفت شود. سپس در خط دوم، تابع `sorted(..., reverse=True)` این جفت‌ها را بر اساس مقدار اولین عنصر هر جفت (یعنی درصد) به‌صورت نزولی مرتب می‌کند. نتیجه این است که خوشه‌ها با توجه به سهمشان در تصویر، از بزرگ‌ترین درصد تا کوچک‌ترین مرتب می‌شوند. در نهایت، `p_and_c` شامل لیستی از جفت‌هایی است که هر کدام سهم یک رنگ و مقدار خود رنگ به‌صورت [B, G, R] را نشان می‌دهد.

ما می توانیم نتیجه حاصل از کد بالا را به صورت یک پالت رنگی نیز نمایش دهیم. به کد زیر دقت کنید

```
block = np.ones((50,50,3),dtype='uint8')
plt.figure(figsize=(12,8))
for i in range(n_clusters):
    plt.subplot(1,n_clusters,i+1)
    block[:] = p_and_c[i][1][::-1] # we have done this to convert bgr(opencv) to
rgb(matplotlib)
    plt.imshow(block)
    plt.xticks([])
    plt.yticks([])
    plt.xlabel(str(round(p_and_c[i][0]*100,2))+'%')
```

این بخش کد برای نمایش رنگ‌های غالب به‌صورت یک پالت تصویری همراه با درصد سهم هر رنگ استفاده می‌شود. در خط اول، `block = np.ones((50,50,3), dtype='uint8')` یک آرایه ۳بعدی با ابعاد ۵۰×۵۰ پیکسل و سه

کانال رنگی می‌سازد که همه مقدارهای آن برابر یک است؛ این آرایه به‌عنوان یک بلاک رنگ برای نمایش استفاده می‌شود. در خط دوم، `plt.figure(figsize=(12,8))` اندازه شکل (Figure) را برای رسم نمودار تنظیم می‌کند. سپس یک حلقه `for`به‌اندازه تعداد خوشه‌ها (`n_clusters`) اجرا می‌شود.

درون حلقه، دستور `plt.subplot(1, n_clusters, i+1)` یک زیرنمودار (Subplot) در یک ردیف با `n_clusters` ستون ایجاد می‌کند و نمودار فعلی را در ستون شماره `i+1` قرار می‌دهد. سپس `block[:] = p_and_c[i][1][::-1]` مقدار همه پیکسل‌های بلوک را برابر با رنگ غالب مربوطه قرار می‌دهد. بعد از آن، `plt.imshow(block)` این بلوک را نمایش می‌دهد. برای حذف اعداد روی محور، `plt.xticks([])` و `plt.yticks([])` استفاده شده است.

در نهایت، `plt.xlabel(str(round(p_and_c[i][0]*100, 2))+'%')` درصد سهم رنگ را با گرد کردن به دو رقم اعشار محاسبه کرده و آن را به‌عنوان برچسب زیر هر بلوک قرار می‌دهد.

خروجی این کد یک ردیف از بلوک‌های رنگی است که از رنگ غالب‌تر به کم‌رنگ‌تر مرتب شده‌اند و درصد هرکدام از تصویر نیز زیر آن‌ها نمایش داده می‌شود.

دلیل استفاده از `i+1`این است که در تابع `plt.subplot` شماره مکان نمودارها **از ۱ شروع می‌شود، نه از ۰**.

به‌طور دقیق، وقتی می‌نویسیم :`plt.subplot(1, n_clusters, position)`

- پارامتر اول (`1`) تعداد ردیف‌ها را مشخص می‌کند.
- پارامتر دوم (`n_clusters`) تعداد ستون‌ها را مشخص می‌کند.
- پارامتر سوم (`position`) جایگاه نمودار فعلی در این شبکه را تعیین می‌کند و این شماره‌گذاری از ۱ آغاز می‌شود.

از آن‌جا که متغیر `i` در حلقه از ۰ تا `n_clusters-1` حرکت می‌کند، اگر به‌طور مستقیم `i` را به‌عنوان مکان استفاده کنیم، در اولین تکرار مقدار صفر خواهد شد که برای `subplot` معتبر نیست. بنابراین `i+1`اضافه می‌کنیم تا شماره‌گذاری از ۱ تا `n_clusters`درست انجام شود.

عبارت `block[:] = p_and_c[i][1][::-1]` در این کد نقش کلیدی در تنظیم رنگ بلوک تصویری دارد و شامل چند بخش مهم است که به تفصیل توضیح داده می‌شود:

- `p_and_c[i]` یک جفت (tuple) شامل دو مقدار است: درصد سهم رنگ (`percentages`) و رنگ غالب متناظر با آن خوشه .(`dominant_colors`) این جفت‌ها قبلاً بر اساس درصد سهم رنگ مرتب شده‌اند.

- `p_and_c[i][1]` عنصر دوم این جفت است که همان آرایه سه‌کاناله رنگ مقادیرB ، G، Rبرای خوشه‌یi ام است.
- `[::-1]` یک عملیات برش (slicing) روی آرایه است که ترتیب عناصر را معکوس می‌کند. در اینجا، این معکوس‌سازی باعث می‌شود ترتیب کانال‌های رنگ از حالت BGR که توسط OpenCV استفاده می‌شود (به حالت RGB که Matplotlibبرای نمایش رنگ‌ها انتظار دارد (تغییر کند.
- `block[:] = ...` واقع به معنای اختصاص مقدار رنگ جدید به تمامی عناصر آرایه‌ی سه‌بعدی `block` است. به بیان دیگر، این دستور تمام پیکسل‌های بلوک ۵۰×۵۰ را با رنگ RGB تبدیل‌شده پر می‌کند تا بلوکی یکنواخت و به رنگ غالب مربوطه ایجاد شود.

در نتیجه، این خط کد با معکوس کردن ترتیب کانال‌های رنگ و پر کردن کامل بلوک با این رنگ، بلوکی از تصویر را به رنگ دقیق هر خوشه‌ی رنگی آماده می‌کند تا در نمودار نمایش داده شود.

دستورات `plt.xticks([])` و `plt.yticks([])` در کتابخانه Matplotlib برای حذف نمایش برچسب‌ها و علامت‌های مربوط به محورهای افقی و عمودی استفاده می‌شوند.

به طور دقیق، `plt.xticks([])` باعث می‌شود که هیچ نشانه‌ای (Tick) و هیچ عدد یا برچسبی در محور افقی نمایش داده نشود و به همین ترتیب، `plt.yticks([])` نشانه‌ها و برچسب‌های محور عمودی را حذف می‌کند.

ما می توانیم این کد را به صورت یک bar chart نیز نمایش دهیم:

```
bar = np.ones((50,500,3),dtype='uint8')
plt.figure(figsize=(12,8))
plt.title('Proportions of colors in the image')
start = 0
i = 1
for p,c in p_and_c:
    end = start+int(p*500)
    if i==n_clusters:
        bar[:,start:] = c[::-1]
    else:
        bar[:,start:end] = c[::-1]
    start = end
    i+=1

plt.imshow(bar)
plt.xticks([])
plt.yticks([])
```

این بخش کد برای ساخت و نمایش یک نوار رنگی طولانی استفاده می‌شود که نشان‌دهنده نسبت هر رنگ غالب در تصویر است. ابتدا با دستور `bar = np.ones((50, 500, 3), dtype='uint8')` یک آرایه سه‌بعدی با ابعاد ۵۰ پیکسل ارتفاع و ۵۰۰ پیکسل عرض ایجاد می‌شود که همه مقادیر آن برابر با یک است؛ این آرایه به‌عنوان بستر نوار رنگی عمل می‌کند. سپس با `plt.figure(figsize=(12,8))` شکل جدیدی برای رسم نمودار با اندازه مناسب ایجاد می‌شود و با `plt.title('Proportions of colors in the image')` عنوانی برای شکل تعیین می‌شود.

در ادامه، متغیر `start` که موقعیت شروع هر بخش رنگ را مشخص می‌کند، صفر قرار داده می‌شود و شمارنده `i` برای پیگیری تعداد خوشه‌ها مقدار اولیه یک می‌گیرد. حلقه `for p, c in p_and_c:` روی جفت‌های درصد سهم (`p`) و رنگ غالب (`c`) اجرا می‌شود. داخل حلقه، متغیر `end` با افزودن تعداد پیکسل‌های متناظر با سهم رنگ محاسبه‌شده به صورت `p * 500` به مقدار `start`به‌روزرسانی می‌شود. سپس با توجه به این که آیا در آخرین خوشه هستیم یا نه، رنگ مربوطه به بخش مناسب نوار اختصاص داده می‌شود. برای اطمینان از تطابق با استاندارد رنگMatplotlib ، ترتیب کانال‌ها با `[::-1]` از BGR به RGB معکوس می‌شود. `p * 500` تعداد پیکسل‌هایی است که آن رنگ باید در طول نوار اشغال کند تا طول بخش رنگی متناسب با درصد واقعی آن در تصویر باشد. بنابراین، این ضرب نقش تبدیل یک نسبت عددی به ابعاد پیکسلی در نوار را ایفا می‌کند.

پس از تخصیص رنگ به بخش مناسب، مقدار `start`به `end` تغییر داده شده و شمارنده `i` افزایش می‌یابد تا در تکرار بعدی بخش بعدی نوار رنگی پر شود. در نهایت، با `plt.imshow(bar)` نوار رنگی نمایش داده می‌شود و دستورات `plt.xticks([])`و `plt.yticks([])`باعث حذف برچسب‌های محورهای افقی و عمودی شده تا تمرکز روی نمایش رنگ‌ها باشد.

نتیجه این کد، یک نوار رنگی افقی است که هر قسمت آن نمایانگر سهم درصدی رنگ غالب مربوطه در تصویر است و به صورت پیوسته رنگ‌ها را از چپ به راست نشان می‌دهد.