Filternig and inrange :

فیلترگذاری روی تصویر به معنای اعمال یک عملیات ریاضی روی پیکسل‌های تصویر است به‌طوری که با استفاده از اطلاعات پیکسل‌های اطراف، ویژگی‌هایی از تصویر **تقویت، حذف یا تغییر** داده شود. این عملیات معمولاً برای **نرم‌کردن، تیزکردن، حذف نویز، یافتن لبه‌ها** و بسیاری کارهای دیگر انجام می‌شود.

در فیلترگذاری، از یک **هسته (Kernel)** یا **ماسک (Mask)** استفاده می‌شود که معمولاً ماتریسی کوچک (مثل 3×3 یا 5×5) است. این هسته روی تمام تصویر حرکت داده می‌شود و با هر ناحیه کوچک از تصویر ضرب شده و جایگزین مقدار پیکسل مرکزی می‌شود.

انواع فیلترها در OpenCV:

1.فیلترهای هموارسازی:(Blurring / Smoothing)

برای **کاهش نویز یا جزئیات اضافی** تصویر استفاده می‌شوند.

- `cv2.blur()`: فیلتر **میانگین‌گیری ساده**
- `cv2.GaussianBlur()`: فیلتر **گوسی** که از وزن‌های متفاوت برای پیکسل‌های اطراف استفاده می‌کند واقع‌گراتر از blur ساده
- `cv2.medianBlur()`: فیلتر **میانه**، بسیار مناسب برای حذف نویز نمکی‌فلفلی

2.فیلترهای تیزکننده:(Sharpening)

برای **تقویت لبه‌ها** و بهتر دیدن جزئیات تصویر به کار می‌روند. معمولاً با استفاده از عملیات دستی با `cv2.filter2D()` و کرنل‌های خاص پیاده‌سازی می‌شوند.

3.فیلترهای تشخیص لبه:(Edge Detection)

برای یافتن **تغییرات ناگهانی روشنایی** یا مرز بین اشیا.

- `cv2.Sobel()`: تشخیص لبه‌ها در جهت x یا y
- `cv2.Laplacian()`: تشخیص لبه‌ها در همه جهت‌ها
- `cv2.Canny()`: یکی از قوی‌ترین الگوریتم‌های تشخیص لبه

4.فیلترهای سفارشی:(Custom Filters)

با استفاده از `cv2.filter2D()` می‌توان هر نوع کرنل دلخواه را تعریف و اعمال کرد.

مثلاً:

- کرنل شناسایی لبه‌ها
- کرنل برجسته‌سازی جزئیات
- کرنل حذف نویز

فیلتر رنگی با **`inRange`** و HSV :

متد `inRange()` در OpenCV ابزاری برای **فیلتر کردن پیکسل‌ها در یک محدوده خاص از مقادیر رنگی یا عددی** است. این متد معمولاً در پردازش تصویر برای **شناسایی رنگ خاص** در تصویر استفاده می‌شود.

```
mask = cv2.inRange(input_image, lower_bound, upper_bound)
```

پارامترها:

- `input_image`: تصویری که می‌خواهیم روی آن فیلتر اعمال کنیم معمولاً تصویر HSV یا Grayscale
- `lower_bound`: آرایه‌ی حد پایین رنگ مورد نظر، مثل `[H_min, S_min, V_min]`.
- `upper_bound`: آرایه‌ی حد بالا برای رنگ مورد نظر، مثل `[H_max, S_max, V_max]`.

خروجی:

- `mask`: یک تصویر **سیاه و سفید** (**تک‌کاناله**) به نام **ماسک**.
  - پیکسل‌هایی که مقدارشان **بین lower و upper باشد**، مقدار **255 (سفید** خواهند داشت.
  - سایر پیکسل‌ها مقدار **0 (سیاه** می‌گیرند.

برای استفاده از عملیات فیلتر گذاری با متد inrange معمولا از فضای رنگی hsv استفاده می شود زیرا:

چون فیلتر کردن رنگ‌ها در فضای HSV **دقیق‌تر، آسان‌تر و قابل کنترل‌تر** است.

درHSV

- **Hueرنگ (**مستقیماً رنگ را مشخص می‌کند (مثلاً قرمز، زرد، آبی)
- در حالی که در BGR یاRGB ، رنگ و روشنایی به هم وابسته‌اند.

بنابراین در این کد:

```
image = cv2.imread('images/felfel-dolme.jpg')

# define range of BLUE color in HSV
lower = np.array([20,50,50])
upper = np.array([35,255,255])

# Convert image from RBG/BGR to HSV so we easily filter
hsv_img = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# Use inRange to capture only the values between lower & upper
mask = cv2.inRange(hsv_img, lower, upper)

#show
plt.figure(figsize=[12,6])
plt.subplot(121);plt.imshow(image[...,::-1]);plt.title("image");
plt.subplot(122);plt.imshow(mask,cmap='gray');plt.title("mask");
```

ابتدا:

محدوده رنگی مورد نظر خود را مشخص کرده ایم. ما در ایم قطعه کد می خواستیم فقط فلفل دلمه ای های زرد رنگ را مشخص کنیم بنابرای ابتدا یک بازه در فضای HSV تعیین کرده و به آن گفته ایم که فقط در این بازه که احتمالا یک بازه مناسب برای رنگ زرد است به جستجوی فلفل دلمه ای های زرد رنگ در تصویر بگردد.

و در ادامه با استفاده از تابع inrange عکس را ماسک کرده و عملیات زیر اتفاق افتاده است:

- با استفاده از تابع `inRange`، از تصویر HSV یک **ماسک سیاه و سفید (mask)** ساخته می‌شود.
- پیکسل‌هایی که مقدارشان بین `lower` و `upper` باشد، در خروجی مقدار **255سفید** می‌گیرند.
- سایر پیکسل‌ها مقدار **0سیاه** خواهند داشت.
- خروجی `mask` یک تصویر تک‌کاناله (grayscale) است.

در هنگام استفاده از فیلتر بر روی عکس های قرمز ممکن است با چالشی روبرو شویم. از آنجایی که رنگ قرمز در فضای HSV در **دو بخش جدا از طیف رنگی** قرار دارد، ابتدا **دو ماسک رنگی** برای بازه‌های پایین و بالای قرمز می سازیم، سپس این دو ماسک با هم **ترکیب** شده و در نهایت، با استفاده از آن یک تصویر جدید ساخته می‌شود که **فقط نواحی قرمز را نمایش می‌دهد** و بقیه‌ی تصویر حذف می‌شود.

برای انجام این عملیات می توانیم از کد زیر استفاده کنیم:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt


image = cv2.imread('images/felfel-dolme.jpg')
# Convert image from RBG/BGR to HSV so we easily filter
hsv_img = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# lower mask (0-10)
lower_red = np.array([0,50,50])
upper_red = np.array([10,255,255])
mask0 = cv2.inRange(hsv_img, lower_red, upper_red)

# upper mask (170-180)
lower_red = np.array([160,50,50])
upper_red = np.array([179,255,255])
mask1 = cv2.inRange(hsv_img, lower_red, upper_red)
```

```
# join masks
mask = mask0+mask1

#result
result = cv2.bitwise_and(image,image, mask=mask)

#show
plt.figure(figsize=[17,7])
plt.subplot(131);plt.imshow(image[...,::-1]);plt.title("image");
plt.subplot(132);plt.imshow(mask, cmap='gray');plt.title("mask");
    plt.subplot(133);plt.imshow(result[...,::-1]);plt.title("result");
```

در این کد:

```
lower_red = np.array([0,50,50])
upper_red = np.array([10,255,255])
```

تعریف بازه‌ی **پایین رنگ قرمز** در فضای HSV.

- Hueبین 0 تا 10 → بخش اول قرمز
- Saturationو Value تنظیم شده‌اند تا شدت رنگ مناسب باشد.

```
mask0 = cv2.inRange(hsv_img, lower_red, upper_red)
```

ساخت ماسک اول:

در این مرحله، پیکسل‌هایی که در این بازه‌ی رنگی هستند، به **(255سفید** (و بقیه به **(0سیاه** (تبدیل می‌شوند.

```
lower_red = np.array([160,50,50])
upper_red = np.array([179,255,255])
```

تعریف بازه‌ی **بالای رنگ قرمز** در Hue بین 160 تا 179.
رنگ قرمز به‌صورت خاص در HSV به‌صورت **دو قسمت جدا در طیف رنگی** قرار دارد، بنابراین باید هر دو بخش را جداگانه بررسی کنیم.

```
mask1 = cv2.inRange(hsv_img, lower_red, upper_red)
```

ساخت ماسک دوم برای بخش دوم رنگ قرمز.
این ماسک نیز تصویر سیاه‌وسفیدی است که فقط نواحی قرمز دوم را شامل می‌شود.

```
mask = mask0 + mask1
```

ترکیب دو ماسک بالا:
همه‌ی پیکسل‌هایی که قرمز هستند (چه از بخش اول، چه دوم) در ماسک نهایی **سفید** خواهند شد. این ماسک نواحی قرمز را به‌طور کامل پوشش می‌دهد.

```
result = cv2.bitwise_and(image,image, mask=mask)
```

اعمال ماسک روی تصویر اصلی:
این خط باعث می‌شود **فقط نواحی قرمز تصویر** باقی بمانند و بقیه‌ی نواحی تصویر سیاه شوند.
در واقع ماسک سفید اجازه عبور پیکسل‌ها را می‌دهد و سیاه جلوی نمایش آن‌ها را می‌گیرد.

در این کد خاص چه اتفاقی می‌افتد؟

1. تابع بررسی می‌کند که کدام پیکسل‌ها در **ماسک مقدار 255 (سفید)** دارند.
2. در آن نقاط، تصویر را **نگه می‌دارد** تصویر × تصویر = خود تصویر.
3. در نقاطی که مقدار ماسک **0سیاه** است، مقدار خروجی را **0سیاه** قرار می‌دهد.