Resize:

تغییر اندازه‌ی تصویر (**Scaling**) یکی از ساده‌ترین و پرکاربردترین تبدیلات هندسی در پردازش تصویر است. در این تبدیل، ابعاد تصویر (عرض و ارتفاع) بزرگ‌تر یا کوچک‌تر می‌شوند، بدون اینکه الزامی برای تغییر شکل کلی تصویر وجود داشته باشد. برای مثال می‌توان یک تصویر کوچک را بزرگ‌نمایی کرد یا یک تصویر بزرگ را برای صرفه‌جویی در حافظه و زمان پردازش کوچک کرد. این عمل بر اساس ضرب مختصات نقاط تصویر در ضرایب مقیاس‌دهی انجام می‌شود.

در سطح ریاضی، تغییر اندازه با یک **ماتریس مقیاس‌دهی** انجام می‌شود که به صورت زیر است:

$$\begin{bmatrix} 0 & 0 & {}_{x}s \\ 0 & {}_{y}s & 0 \end{bmatrix} = S$$

در این ماتریس، xs ضریب بزرگ‌نمایی یا کوچک‌نمایی در محور افقی (x) و ys ضریب تغییر اندازه در محور عمودی (y) هستند. اگر این ضرایب بزرگ‌تر از ۱ باشند، تصویر بزرگ‌تر می‌شود و اگر بین ۰ و ۱ باشند، تصویر کوچک‌تر خواهد شد.

در کتابخانه‌ی OpenCV ، تغییر اندازه معمولاً با تابع `cv2.resize` انجام می‌شود. این تابع علاوه بر ابعاد جدید، یک روش درون‌یابی (Interpolation) هم دریافت می‌کند تا مقادیر پیکسل‌های جدید تعیین شوند. روش‌هایی مثل **Nearest Neighbor** (ساده و سریع(، **Bilinear**(کیفیت بهتر) و **Bicubic**(کیفیت بالاتر برای بزرگ‌نمایی) وجود دارند. این انتخاب باعث می‌شود تصویر نهایی بسته به نیاز یا کیفیت مطلوب تغییر کند. تغییر اندازه در کاربردهایی مثل آماده‌سازی داده برای شبکه‌های عصبی، فشرده‌سازی تصویر و نمایش تصاویر روی دستگاه‌های مختلف بسیار پرکاربرد است.

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

image = cv2.imread('images/input.jpg')

# Store height and width of the image
height, width = image.shape[:2]

#       | Sx 0 0 |
#  T  = | 0 Sy 0 |

# T is our translation matrix
T = np.float32([[0.5, 0, 0], [0, 0.5,0]])
```

```
# We use warpAffine to transform the image using the matrix, T
img_resized = cv2.warpAffine(image, T, (width//2, height//2))


plt.figure(figsize=[12,7])
plt.subplot(121);plt.imshow(image[...,::-1]);plt.title("image");
plt.subplot(122);plt.imshow(img_resized[...,::-1]);plt.title("resized");
```

این کد ابتدا یک تصویر را با استفاده از `cv2.imread`بارگذاری می‌کند و سپس ابعاد آن (ارتفاع و عرض) را به دست می‌آورد. بخش بعدی مربوط به تعریف یک ماتریس تغییر اندازه (Scaling Matrix) است. در این ماتریس، مقادیر Sx=0.5و Sy=0.5در نظر گرفته شده‌اند، به این معنا که تصویر در هر دو محور افقی و عمودی نصف می‌شود. در نتیجه، خروجی نهایی نسبت به تصویر اصلی کوچک‌تر خواهد بود.

برای اعمال این تغییر اندازه، از تابع `cv2.warpAffine`استفاده می‌شود. این تابع تصویر اصلی، ماتریس تبدیل T و ابعاد جدید تصویر را می‌گیرد. در اینجا، اندازه خروجی `(width//2, height//2)` قرار داده شده که دقیقاً نصف ابعاد اولیه است. پس تصویر خروجی (`img_resized`) نسخه‌ای کوچک‌تر از تصویر ورودی خواهد بود. لازم به ذکر است که چون از ماتریس آفین استفاده شده، مقیاس‌دهی با دقت محاسباتی بر اساس مقادیر سینوس و کسینوس یا ضرایب داده شده انجام می‌گیرد.

در پایان کد، از `matplotlib.pyplot` برای نمایش همزمان تصویر اصلی و تصویر تغییر اندازه داده‌شده استفاده شده است. با دستور `plt.subplot`دو تصویر در کنار هم قرار داده می‌شوند: سمت چپ تصویر اصلی (`image`) و سمت راست تصویر کوچک‌شده .(`img_resized`) همچنین، برای اینکه رنگ‌ها به درستی نمایش داده شوند، کانال‌های رنگی از BGR)که خروجی پیش‌فرض OpenCV است) به RGB فرمت مورد انتظار (matplotlib تغییر داده شده‌اند. به این ترتیب، کاربر می‌تواند مقایسه‌ای مستقیم بین تصویر اولیه و تصویر تغییر اندازه داده‌شده داشته باشد.

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

rotation_amount_degree = 10
#  convert rotation amount to radian
theta = rotation_amount_degree * np.pi / 180.0

image = cv2.imread('images/input.jpg')
height, width, _ = image.shape
T1 = np.float32([[np.cos(theta), -np.sin(theta)], [np.sin(theta), np.cos(theta)]]) #Rotate
```

```
T2 = np.float32([[0.5, 0], [0, 0.5]]) #Scale
T = np.matmul(T1,T2)
final_T = np.ones((2,3))*50 #Rotate + Scale
final_T[:,:-1] = T
print(final_T)
```

این کد یک گام جلوتر از مثال قبلی است و **چرخش و تغییر اندازه تصویر را به صورت ترکیبی** انجام می‌دهد. ابتدا زاویه‌ی چرخش `rotation_amount_degree = 10`درجه تعریف شده و با ضرب در $\pi/180$به رادیان تبدیل می‌شود (`theta`)تا بتوان از آن در محاسبات ماتریس چرخش استفاده کرد. تصویر ورودی با `cv2.imread`خوانده شده و ابعاد آن (ارتفاع و عرض) استخراج می‌شود تا برای محاسبات بعدی و تعیین اندازه خروجی استفاده شوند.

در مرحله‌ی بعد، دو ماتریس تعریف می‌شوند:

1. `T1` → ماتریس **چرخش**، که بر اساس سینوس و کسینوس زاویه θساخته شده است.
2. `T2` → ماتریس **مقیاس‌دهی**، که با ضرایب 0.5 در هر دو محور تصویر را نصف می‌کند.

سپس با دستور `np.matmul(T1, T2)` این دو ماتریس با هم ضرب می‌شوند تا یک ماتریس ترکیبی حاصل شود که همزمان چرخش و کوچک‌سازی تصویر را انجام دهد.

برای آماده‌سازی ماتریس نهایی قابل استفاده در `cv2.warpAffine`، یک ماتریس `2×3`به نام `final_T`ساخته می‌شود و بخش ۲×۲ آن با ماتریس ترکیبی جایگزین می‌شود و ستون سوم (جابجایی) برابر 50 قرار داده می‌شود. در نهایت، تابع `cv2.warpAffine`تصویر ورودی را با ماتریس `final_T`تبدیل می‌کند و خروجی (`result`) شامل تصویر چرخیده و کوچک‌شده همراه با جابجایی است. با استفاده از `matplotlib.pyplot`، تصویر اصلی و نتیجه نهایی در کنار هم نمایش داده می‌شوند.

تفاوت با کد قبلی:

- در کد قبلی فقط **تغییر اندازه** (Scaling)انجام می‌شد، بدون چرخش و جابجایی.
- در این کد، همزمان **چرخش** و **مقیاس‌دهی** ترکیب شده‌اند و علاوه بر آن یک مقدار **جابجایی (Translation)** نیز اعمال شده است.
- همچنین در این کد از ضرب ماتریسی (`np.matmul`) برای ترکیب تبدیلات استفاده شده، در حالی که در مثال قبل تنها یک ماتریس ساده‌ی ۲×۳ برای Scaling ساخته شده بود.

```python
import cv2
import numpy as np

# load our input image
image = cv2.imread('images/input.jpg')
cv2.imshow('Original Image', image)
cv2.waitKey()

# Let's make our image 3/4 of it's original size
image_scaled = cv2.resize(image, None, fx=0.75, fy=0.75)
cv2.imshow('Scaling - Linear Interpolation', image_scaled)
cv2.waitKey()

# Let's double the size of our image
img_scaled = cv2.resize(image, None, fx=2, fy=2, interpolation = cv2.INTER_CUBIC)
cv2.imshow('Scaling - Cubic Interpolation', img_scaled)
cv2.waitKey()

# Let's skew the re-sizing by setting exact dimensions
img_scaled = cv2.resize(image, (900, 400), interpolation = cv2.INTER_AREA)
cv2.imshow('Scaling - Skewed Size', img_scaled)
cv2.waitKey()

cv2.destroyAllWindows()
```

این کد مربوط به **تغییر اندازه تصاویر (Scaling / Resizing)** با استفاده از کتابخانه‌ی OpenCV است و نحوه‌ی استفاده از روش‌های مختلف درون‌یابی را برای کوچک‌کردن یا بزرگ‌کردن تصویر نشان می‌دهد. ابتدا تصویر ورودی با تابع `cv2.imread` خوانده می‌شود و با `cv2.imshow` در یک پنجره نمایش داده می‌شود. تابع `cv2.waitKey()` باعث می‌شود که نمایش تصویر متوقف شود تا کاربر کلیدی را فشار دهد و پنجره بسته نشود. این مرحله به کاربر اجازه می‌دهد تصویر اصلی را قبل از اعمال تغییرات مشاهده کند.

در بخش بعد، تصویر با استفاده از `cv2.resize` به اندازه‌ی **۴/۳ اندازه‌ی اصلی** کاهش داده می‌شود. ضرایب `fx=0.75` و `fy=0.75` تعیین‌کننده‌ی مقیاس در محورهای افقی و عمودی هستند. روش پیش‌فرض در این حالت **Linear Interpolation** است که برای کاهش اندازه تصویر کیفیت مناسبی ارائه می‌دهد و لبه‌ها را نسبتا صاف نگه می‌دارد. تصویر کوچک‌شده سپس با یک پنجره جداگانه نمایش داده می‌شود.

در مرحله‌ی بعد، تصویر دوباره تغییر اندازه داده می‌شود، اما این بار با **دو برابر اندازه‌ی اصلی** و با استفاده از روش **Cubic Interpolation**. این روش برای بزرگ‌نمایی تصاویر بهتر از Linear عمل می‌کند، زیرا با محاسبه‌ی مقادیر پیکسل‌های جدید بر اساس چندین پیکسل مجاور، کیفیت و نرمی لبه‌ها حفظ می‌شود. به این ترتیب تصویر بزرگ‌شده واضح و طبیعی به نظر می‌رسد.

در نهایت، تصویر به ابعاد مشخص `(900, 400)` تغییر اندازه داده می‌شود. در این حالت نسبت ابعاد تصویر اصلی حفظ نمی‌شود و تصویر ممکن است کشیده یا فشرده شود. برای این نوع تغییر اندازه از روش `INTER_AREA` استفاده شده که کیفیت خوبی هنگام کوچک کردن تصویر ارائه می‌دهد. در پایان، با استفاده از `cv2.destroyAllWindows()` تمام پنجره‌های باز شده توسط OpenCV بسته می‌شوند تا برنامه به درستی پایان یابد. به طور خلاصه، این کد سه روش تغییر اندازه تصویر را نمایش می‌دهد: کوچک‌کردن نسبی، بزرگ‌کردن با کیفیت بالا و تغییر اندازه با ابعاد مشخص.