Seamless-clone :

در کتابخانه **OpenCV** قابلیتی به نام **Seamless Cloning** وجود دارد که برای ادغام یک ناحیه از تصویر منبع در تصویر مقصد به‌کار می‌رود، به‌طوری‌که مرز بین آن‌ها طبیعی و بدون درز (seamless) به نظر برسد. این روش به کمک الگوریتم‌های پردازش تصویر مانند *Poisson Image Editing* عمل می‌کند و هدفش این است که ناحیه کپی‌شده از نظر رنگ، نور و بافت با تصویر پس‌زمینه هماهنگ شود. این قابلیت در بسیاری از کاربردها مانند ویرایش عکس، فتومونتاژ، گرافیک کامپیوتری و حتی در پروژه‌های واقعیت افزوده به‌کار می‌رود.

برای پیاده‌سازی seamless cloning در OpenCV از تابع زیر استفاده می‌شود:

```
cv2.seamlessClone(src, dst, mask, center, flags)
```

- `Src` تصویر منبع (بخشی که می‌خواهیم کپی کنیم).
- `Dst` تصویر مقصد (محل قرارگیری ناحیه جدید).
- `Mask` ناحیه‌ای که باید کپی شود (معمولاً یک ماسک سیاه و سفید).
- `Center` مختصات مرکز قرارگیری ناحیه در تصویر مقصد.
- `Flags` مشخص‌کننده نوع مدل ترکیب.(Blending Model)

در OpenCV سه مدل مختلف برای seamless clone وجود دارد. مدل **NORMAL_CLONE** که رایج‌ترین حالت است، سعی می‌کند ناحیه منبع را با در نظر گرفتن رنگ‌ها و بافت‌های تصویر مقصد ترکیب کند و نتیجه‌ای طبیعی ایجاد کند. این حالت برای جایگذاری اشیاء یا تغییر پس‌زمینه‌ها بسیار مناسب است.

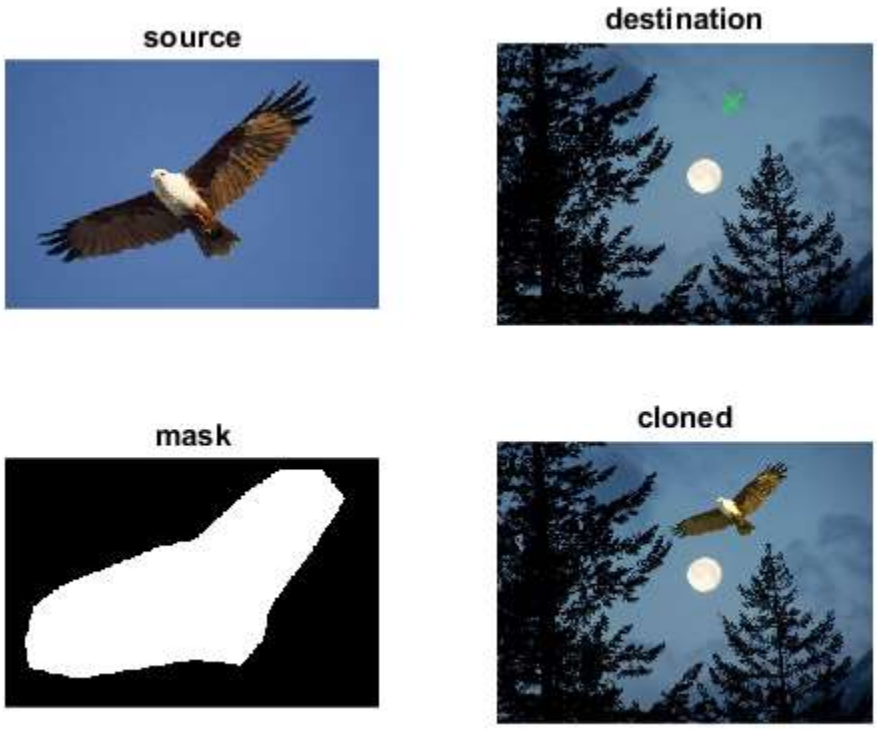
مدل دوم **MIXED_CLONE** است. این روش علاوه بر در نظر گرفتن رنگ‌ها، تغییرات گرادیان (تفاوت‌های شدت روشنایی) را نیز بررسی می‌کند و اجازه می‌دهد جزئیات بیشتری از تصویر منبع حفظ شود. این مدل معمولاً زمانی استفاده می‌شود که بخواهیم بافت‌ها یا جزئیات لبه‌های ناحیه منبع به خوبی منتقل شوند. مدل سوم **MONOCHROME_TRANSFER** نام دارد که بیشتر روی انتقال تونالیته (رنگ‌مایه کلی یا روشنایی) از مقصد به منبع تمرکز می‌کند و در شرایطی که بخواهیم هماهنگی رنگی ساده و سریع داشته باشیم به کار می‌رود.

مراحل پیاده سازی:

برای پیاده‌سازی **seamless cloning** در OpenCV ، ابتدا باید دو تصویر داشته باشیم: تصویر **منبع (src)** که بخشی از آن قرار است به تصویر دیگر منتقل شود، و تصویر **مقصد (dst)** که قرار است شیء در آن قرار بگیرد. سپس یک **ماسک (mask)** ساخته می‌شود که دقیقاً ناحیه موردنظر از تصویر منبع را مشخص می‌کند؛ این ماسک می‌تواند دستی (مثلاً با چندضلعی یا

مستطیل) یا خودکار) مثلاً با threshold یا کانتور (ساخته شود. همچنین باید یک نقطه‌ی **مرکز (center)** مشخص شود که تعیین می‌کند ناحیه انتخاب‌شده در چه مختصاتی از تصویر مقصد قرار گیرد.

پس از آماده‌سازی این ورودی‌ها، از تابع `cv2.seamlessClone(src, dst, mask, center, flags)` استفاده می‌کنیم. در این تابع، `flags` مشخص می‌کند از کدام مدل ترکیب استفاده شود مانند `NORMAL_CLONE`، `MIXED_CLONE` یا `MONOCHROME_TRANSFER` نتیجه‌ی این تابع تصویری است که ناحیه‌ی انتخاب‌شده از منبع، به شکلی طبیعی و بدون مرز در تصویر مقصد ترکیب شده است. در نهایت خروجی را می‌توان با `cv2.imshow` یا `matplotlib` نمایش داد یا در قالب یک فایل تصویری ذخیره کرد.



چگونه یک ماسک بسازیم؟

```
src = cv2.imread("images/airplane.jpg")
poly = np.array([ [4,80], [30,54], [151,63], [254,37], [298,90], [272,134], [43,122] ],
np.int32)
cv2.polylines(src, [poly],1, (0, 0, 255))

src_mask = np.zeros(src.shape, src.dtype)
cv2.fillPoly(src_mask, [poly], (255, 255, 255))
```

این کد ابتدا تصویر منبع (airplane.jpg) را بارگذاری می‌کند و یک چندضلعی (polygon) با مختصات مشخص تعریف می‌کند. این چندضلعی در واقع ناحیه‌ای از تصویر را که قرار است انتخاب شود مشخص می‌کند. سپس با استفاده از دستور cv2.polylines خطوط چندضلعی روی تصویر اصلی کشیده می‌شود تا محدوده‌ی انتخاب‌شده به صورت بصری مشخص گردد. رنگ خطها قرمز است چون مقدار (0, 0, 255) به عنوان رنگ داده شده است فرمت رنگ در OpenCV به صورت BGR است.

در ادامه، یک تصویر ماسک سیاه به اندازه‌ی تصویر اصلی ساخته می‌شود .(src_mask) سپس با دستور cv2.fillPoly همان چندضلعی روی این ماسک پر می‌شود، طوری که در ناحیه‌ی چندضلعی مقدار پیکسل‌ها سفید (255,255,255) و در باقی قسمت‌ها سیاه (0,0,0) باقی می‌ماند. این ماسک بعداً می‌تواند برای جداسازی ناحیه موردنظر یا برای عملیات‌هایی مثل seamless cloning استفاده شود.

در بخش پایانی، با استفاده از matplotlib دو تصویر در کنار هم نمایش داده می‌شوند. تصویر سمت چپ (subplot(121))همان تصویر اصلی است که روی آن چندضلعی قرمز ترسیم شده است، و تصویر سمت راست (subplot(122))همان ماسکی است که فقط ناحیه‌ی انتخاب‌شده (چندضلعی سفید روی زمینه‌ی سیاه) را نشان می‌دهد. این کار کمک می‌کند تا هم ناحیه انتخاب‌شده روی تصویر دیده شود و هم ماسک نهایی برای استفاده در مراحل بعدی آماده باشد.

مرحله بعد نوبت به پیاده سازی است:

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

# Read images
src = cv2.imread("images/airplane.jpg")
dst = cv2.imread("images/input.jpg")

# Create a rough mask around the airplane.
src_mask = np.zeros(src.shape, src.dtype)
poly = np.array([ [4,80], [30,54], [151,63], [254,37], [298,90], [272,134], [43,122] ],
np.int32)
cv2.fillPoly(src_mask, [poly], (255, 255, 255))

# This is where the CENTER of the airplane will be placed
center = (200,100)

# Clone seamlessly.
output = cv2.seamlessClone(src, dst, src_mask, center, cv2.NORMAL_CLONE)
```

```
plt.figure(figsize=[12,7])
plt.imshow(output[...,::-1])
```

این کد یک نمونه کامل از **Seamless Cloning** در OpenCV است که در آن یک شیء (هواپیما) از یک تصویر منبع به تصویر مقصد منتقل می‌شود به طوری که مرزهای آن طبیعی به نظر برسند. ابتدا دو تصویر بارگذاری می‌شوند: تصویر منبع (`src`) که شامل هواپیما است و تصویر مقصد (`dst`) که قرار است هواپیما در آن قرار گیرد.

در مرحله بعد، یک **ماسک خالی** به اندازه‌ی تصویر منبع ساخته می‌شود (`src_mask = np.zeros(...)`) و یک چندضلعی (`poly`) تعریف می‌شود که ناحیه تقریبی هواپیما را مشخص می‌کند. سپس با استفاده از دستور `cv2.fillPoly` این چندضلعی روی ماسک پر می‌شود تا قسمت سفید آن ناحیه‌ای باشد که قرار است به تصویر مقصد منتقل شود، و قسمت‌های سیاه باقی‌مانده ناحیه‌های غیرانتقالی هستند. این ماسک، محدوده‌ای از تصویر منبع را که باید کپی شود تعیین می‌کند.

در ادامه، مختصات **مرکز قرارگیری** شیء در تصویر مقصد با متغیر `center` تعیین می‌شود. سپس تابع `cv2.seamlessClone` اجرا می‌شود تا ناحیه مشخص‌شده توسط ماسک از تصویر منبع به تصویر مقصد منتقل شود و یک تصویر خروجی با ترکیب طبیعی ایجاد گردد. در نهایت، تصویر نهایی با استفاده از matplotlib نمایش داده می‌شود؛ `[...,::-1]` رنگ‌ها را از BGR (OpenCV) به RGB (matplotlib) تبدیل می‌کند تا رنگ‌ها درست نمایش داده شوند.