GrabCut:

الگوریتم **GrabCu t** یکی از روش‌های پرکاربرد در پردازش تصویر برای جداسازی سوژه (Foreground) از پس‌زمینه (Background)است. این الگوریتم در سال ۲۰۰۴ توسط مایکروسافت معرفی شد و هدف اصلی آن این بود که با حداقل تعامل کاربر، جداسازی اجسام در تصویر به‌طور دقیق انجام شود. ایده اصلی بر پایه‌مدل‌سازی آماری رنگ‌ها با استفاده از **مدل گوسی (GMM)** و سپس بهینه‌سازی بر اساس الگوریتم **Graph Cut** شکل گرفته است.

در عمل، کاربر معمولاً با کشیدن یک مستطیل دور سوژه، ناحیه تقریبی جسم را مشخص می‌کند. الگوریتم سپس با بررسی پیکسل‌های داخل مستطیل، آن‌ها را به‌عنوان احتمالی از "پیش‌زمینه" و پیکسل‌های بیرون مستطیل را به‌عنوان "پس‌زمینه" در نظر می‌گیرد. این برچسب‌گذاری اولیه پایه کار را تشکیل می‌دهد، و سپس GrabCut به‌صورت تکراری (iterative) مدل‌های آماری رنگ‌ها را یاد می‌گیرد و بهبود می‌دهد.

الگوریتم از مفهوم **گراف** استفاده می‌کند؛ هر پیکسل به‌عنوان یک گره در نظر گرفته می‌شود و یال‌ها (Edges) ارتباط بین پیکسل‌ها و احتمال تعلق به پیش‌زمینه یا پس‌زمینه را نشان می‌دهند. با استفاده از تکنیک **Minimum Cut**، الگوریتم مرزی را پیدا می‌کند که کمترین هزینه جداسازی را دارد، یعنی مرزی که بهترین تفکیک بین جسم و پس‌زمینه را ارائه می‌دهد. این فرآیند در چندین تکرار انجام می‌شود تا نتیجه به‌طور تدریجی دقیق‌تر گردد.

مزیت اصلی GrabCut در این است که نیاز به دخالت کم کاربر دارد و معمولاً کیفیت جداسازی بالایی به دست می‌دهد. علاوه بر این، امکان اصلاح دستی هم وجود دارد؛ یعنی کاربر می‌تواند بخش‌هایی را که اشتباه برچسب‌گذاری شده‌اند مشخص کند و الگوریتم دوباره محاسبات خود را اصلاح نماید. همین ویژگی باعث شده GrabCut در نرم‌افزارهای ویرایش تصویر مثل Photoshop یا GIMP و همچنین پروژه‌های بینایی ماشین و بینایی کامپیوتر کاربرد زیادی داشته باشد.

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

img = cv2.imread('images/pic.png)
mask = np.zeros(img.shape[:2],np.uint8)
bgdModel = np.zeros((1,65),np.float64)
fgdModel = np.zeros((1,65),np.float64)
rect = (21,50,460,600)
cv2.grabCut(img,mask,rect,bgdModel,fgdModel,5,cv2.GC_INIT_WITH_RECT)
mask2 = np.where((mask==2)|(mask==0),0,1).astype('uint8')
result = img*mask2[:,:,np.newaxis]
```

```
plt.figure(figsize=[10,5])
plt.subplot(121);plt.imshow(img[...,::-1]);plt.title("image");
plt.subplot(122);plt.imshow(result[...,::-1]);plt.title("result");
```

بیایید این کد را با هم بررسی کنیم

```
mask = np.zeros(img.shape[:2],np.uint8)
bgdModel = np.zeros((1,65),np.float64)
fgdModel = np.zeros((1,65),np.float64)
```

- `img.shape[:2]` فقط ارتفاع و عرض تصویر را می‌گیرد (بدون کانال‌های رنگی).
- یک ماتریس صفر با همین ابعاد ساخته می‌شود. نوع داده هم `uint8`(اعداد صحیح ۸ بیتی) است.
- این **ماسک** جایی است که الگوریتم GrabCut در هر پیکسل مشخص می‌کند آیا آن پیکسل به پس‌زمینه یا پیش‌زمینه تعلق دارد.
- مقادیر ماسک می‌توانند یکی از این چهار حالت باشند:
  - 0 → پس‌زمینه قطعی(GC_BGD)
  - 1 → پیش‌زمینه قطعی(GC_FGD)
  - 2 → پس‌زمینه احتمالی(GC_PR_BGD)
  - 3 → پیش‌زمینه احتمالی(GC_PR_FGD)

مدل پس زمینه:

این آرایه یک فضای خالی برای ذخیره‌ی مدل آماری **پس‌زمینه** است.

الگوریتم GrabCut از **مدل‌های مخلوط گاوسی (GMM)** برای یادگیری رنگ‌ها استفاده می‌کند.

ابعاد `(1,65)` و نوع داده `float64`توسط OpenCV از پیش تعریف شده‌اند. برنامه‌نویس مستقیماً این مقادیر را تغییر نمی‌دهد، بلکه GrabCut آن‌ها را پر می‌کند.

مدل پیش زمینه:

مشابه `bgdModel`، این آرایه برای ذخیره‌ی مدل آماری **پیش‌زمینه** استفاده می‌شود.

در طول اجرایGrabCut ، این مدل به‌تدریج از داده‌های تصویری یاد می‌گیرد که چه رنگ‌هایی بیشتر مربوط به شیء اصلی هستند.

```
rect = (21,50,460,600)
cv2.grabCut(img,mask,rect,bgdModel,fgdModel,5,cv2.GC_INIT_WITH_RECT)
```

`rect` محدوده‌ای تقریبی از جایی است که سوژه در آن قرار دارد ,x, y عرض، ارتفاع

`cv2.grabCut` اجرا می‌شود:

- `img`تصویر ورودی.
- `mask`ماسک اولیه (الگوریتم در طول کار آن را پر می‌کند).
- `rect`محدوده‌ای که شامل سوژه است.
- `bgdModel, fgdModel`مدل‌های آماری که در طول کار پر می‌شوند.
- 5تعداد تکرار الگوریتم.
- `cv2.GC_INIT_WITH_RECT` الگوریتم بداند شروع از مستطیل باشد.

```
mask2 = np.where((mask==2)|(mask==0),0,1).astype('uint8')
result = img*mask2[:,:,np.newaxis]
```

در این بخش ابتدا با خط

```
mask2 = np.where((mask==2)|(mask==0),0,1).astype('uint8')
```

خروجی ماسک تولیدشده توسط الگوریتم GrabCut به یک ماسک ساده باینری تبدیل می‌شود. ماسک اصلی شامل چهار مقدار مختلف است (۰: پس‌زمینه قطعی، ۱: پیش‌زمینه قطعی، ۲: پس‌زمینه احتمالی، ۳: پیش‌زمینه احتمالی). در این خط، هر پیکسل که مقدارش ۰ یا ۲ باشد به‌عنوان پس‌زمینه در نظر گرفته شده و صفر می‌شود؛ بقیه مقادیر (۱ و ۳) برابر یک قرار می‌گیرند. نتیجه، یک آرایه دوبعدی با مقادیر ۰ و ۱ است که به‌صورت ماسک نهایی برای تفکیک پیش‌زمینه از پس‌زمینه عمل می‌کند.

سپس در خط

```
result = img*mask2[:,:,np.newaxis]
```

این ماسک روی تصویر اصلی اعمال می‌شود. چون `mask2` دوبعدی است، با `np.newaxis` یک بُعد به آن اضافه می‌شود تا هم‌راستا با سه کانال رنگی تصویر باشد. حالا ضرب عنصر به عنصر انجام می‌شود: هر پیکسل که ماسکش صفر است (پس‌زمینه)، در خروجی سیاه می‌شود؛ و هر پیکسل که ماسکش یک است (پیش‌زمینه)، همان مقدار رنگ اصلی تصویر را حفظ می‌کند. بنابراین خروجی `result` تصویری است که فقط سوژه اصلی را نمایش می‌دهد و پس‌زمینه حذف شده است.

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

img = cv2.imread('images/akhavan.jpg')
bgdmodel = np.zeros((1, 65), np.float64)
fgdmodel = np.zeros((1, 65), np.float64)
cv2.grabCut(img,mask,None,bgdModel,fgdModel,5,cv2.GC_INIT_WITH_MASK)

mask2 = np.where((mask==cv2.GC_PR_BGD)|(mask==cv2.GC_BGD),0,1).astype('uint8')
result = img*mask2[:,:,np.newaxis]

plt.figure(figsize=[10,5])
plt.subplot(121);plt.imshow(img[...,::-1]);plt.title("image");
plt.subplot(122);plt.imshow(result[...,::-1]);plt.title("result");
```

در این کد، تصویر با استفاده از `cv2.imread` خوانده می‌شود و مانند قبل دو مدل آماری `bgdModel` و `fgdModel` برای ذخیره اطلاعات مربوط به پس‌زمینه و پیش‌زمینه ایجاد می‌گردند. تفاوت مهم اینجاست که به جای تعریف یک مستطیل اولیه، از ماسک آماده به‌عنوان ورودی استفاده شده است. تابع `cv2.grabCut` در این حالت با پارامتر `cv2.GC_INIT_WITH_MASK` اجرا می‌شود، بنابراین الگوریتم جداسازی را بر اساس برچسب‌های موجود در ماسک آغاز می‌کند. این برچسب‌ها معمولاً توسط کاربر یا یک مرحله‌ی پیش‌پردازش تعیین می‌شوند و به الگوریتم کمک می‌کنند از همان ابتدا مرزهای دقیق‌تری بین سوژه و پس‌زمینه داشته باشد.

پس از اجرای GrabCut ، مقادیر ماسک خروجی بررسی و ساده‌سازی می‌شوند. در خط

```
mask2 = np.where((mask==cv2.GC_PR_BGD)|(mask==cv2.GC_BGD),0,1).astype('uint8')
```

پیکسل‌هایی که به‌عنوان پس‌زمینه‌ی قطعی یا احتمالی برچسب خورده‌اند صفر می‌شوند و پیکسل‌های پیش‌زمینه‌ی قطعی یا احتمالی برابر یک قرار می‌گیرند. سپس با ضرب کردن تصویر اصلی در این ماسک سه‌بعدی، بخش پس‌زمینه حذف و تنها سوژه باقی می‌ماند. در انتها با استفاده از Matplotlib تصویر اولیه و نتیجه‌ی پردازش در کنار یکدیگر نمایش داده می‌شوند تا تفاوت آشکار گردد.