حذف پرده سبز:

پرده سبز یا **صفحه سبز** تکنیکی در حوزه‌ی **تصویربرداری، فیلم‌سازی، گرافیک کامپیوتری** و حتی **پخش زنده** است که به کمک آن می‌توان پس‌زمینه‌ی تصویر را حذف یا با تصویر/ویدئویی دیگر جایگزین کرد.

این تکنیک به‌طور رسمی‌تر با نام **کِی‌ینگ (Chroma Keying)** شناخته می‌شود.

- **رنگ سبز** معمولاً انتخاب می‌شود چون:
  - بیشترین **تضاد** را با رنگ پوست انسان دارد.
  - در لباس‌ها و بدن انسان‌ها به ندرت یافت می‌شود.
  - نسبت به رنگ آبی، در حسگرهای دوربین **نور بیشتری ثبت می‌شود**)بهتر دیده می‌شود.(

)گاهی در موارد خاص از پرده **آبی** استفاده می‌شود، مثلاً وقتی سوژه لباس سبز دارد(.

نحوه عملکرد پرده سبز:

1. **فیلم‌برداری یا عکاسی** از سوژه (مثلاً یک فرد) در جلوی پرده‌ی سبز انجام می‌شود.
2. در مرحله‌ی پردازش (با نرم‌افزار یا کدنویسی)، رنگ سبز به‌عنوان رنگ **پس‌زمینه** شناسایی می‌شود.
3. پیکسل‌هایی که **مقدار رنگ آن‌ها در محدوده سبز قرار دارد، حذف می‌شوند** یا **شفاف (transparent)** در نظر گرفته می‌شوند.
4. سپس، تصویر یا ویدئوی دلخواه به‌جای آن پس‌زمینه قرار می‌گیرد.
5. •در برنامه‌نویسی، معمولاً تصویر به **فرمت HSV** تبدیل می‌شود.
6. •سپس محدوده‌ای از رنگ سبز Hue حدود 35 تا 85 مشخص می‌شود.
7. •ماسک رنگ ساخته می‌شود (پیکسل‌هایی که داخل این بازه هستند).
8. •این ماسک برای حذف، شفاف‌سازی یا جایگزینی پس‌زمینه استفاده می‌شود.

برای حذف پرده سبز از یک تصویر باید مقدار رنگ سبز در HSV را مشخص کنیم. برای این کار میتوانیم از کد زیر استفاده کنیم

```
input_image = cv2.imread("images/green-screen.png")
croped_region = input_image[:50,:50,:]
plt.imshow(croped_region[...,::-1])

hsv = cv2.cvtColor(croped_region, cv2.COLOR_BGR2HSV)
```

```
print("HSV: ", np.mean(hsv, axis=(0,1)))
```

این کد با استفاده از کتابخانه‌های **OpenCV** و **Matplotlib**، بخشی از یک تصویر را که شامل پرده سبز است تحلیل می‌کند تا **میانگین رنگ در فضای رنگی HSV** آن ناحیه را به دست آورد. کاربرد این کد معمولاً در پروژه‌هایی است که می‌خواهیم پرده‌ی سبز را شناسایی و حذف کنیم .(chroma keying) حال بیایید مراحل آن را به‌صورت متوالی و مفهومی شرح دهیم:

پس از خواندن تصویر در این مرحله، یک **ناحیه‌ی مربعی به ابعاد ۵۰ در ۵۰ پیکسل** از گوشه‌ی بالای چپ تصویر (ردیف‌ها و ستون‌های ۰ تا ۴۹) انتخاب و برش داده می‌شود. این ناحیه معمولاً بخشی از پس‌زمینه‌ی سبز است که در تکنیک پرده‌ی سبز مورد نظر ماست. با انتخاب این ناحیه، می‌خواهیم مشخصات رنگ سبزِ موجود در تصویر را بررسی کنیم. سپس ناحیه‌ی انتخاب‌شده از فضای رنگی BGR به **HSV (Hue, Saturation, Value)**تبدیل می‌شود. این فضا نسبت به تغییرات نور و روشنایی مقاوم‌تر است و در پردازش تصویر، مخصوصاً برای **تشخیص رنگ‌ها (مانند رنگ سبز پرده)**، بسیار مفیدتر از BGR یا RGB است.

در نهایت مقادیر HSV تمام پیکسل‌های ناحیه‌ی برش‌خورده را بررسی می‌کنیم. با استفاده از تابع `np.mean()` و مشخص کردن محورهای `(0, 1)`، میانگین مقدار سه کانال Hue، Saturation، Value) در کل ناحیه‌ی ۵۰×۵۰ محاسبه می‌شود. این میانگین به ما کمک می‌کند تا **دامنه‌ی عددی رنگ سبز موجود در تصویر** را به‌طور دقیق به دست آوریم، که گام مهمی در ساخت **ماسک پرده سبز و حذف پس‌زمینه** است.

اکنون که مقدار پیکسل رنگ سبز مشخص شد می توانیم با استفاده از کد زیر پرده سبز را حذف کنیم:

```
input_image = cv2.imread("images/green-screen.png")
bg_image = cv2.imread("images/balloon.png")
hsv = cv2.cvtColor(input_image, cv2.COLOR_BGR2HSV)

h, w, _ =input_image.shape
bg_image = cv2.resize(bg_image,(w, h))
l_green = np.array([35, 50, 50])
u_green = np.array([85, 255, 255])

mask = cv2.inRange(hsv, l_green, u_green)
mask_not_green = cv2.bitwise_not(mask)

new_bg = cv2.bitwise_and(bg_image, bg_image, mask=mask)
removed_bg = cv2.bitwise_and(input_image, input_image, mask=mask_not_green)

final = cv2.add(removed_bg, new_bg)

plt.figure(figsize=[12,7])
```

```
plt.subplot(231);plt.imshow(input_image[...,::-1]);plt.title("Original");
plt.subplot(232);plt.imshow(mask, cmap='gray');plt.title("Green mask");
plt.subplot(233);plt.imshow(mask_not_green, cmap='gray');plt.title("Not green mask");
#second row
plt.subplot(234);plt.imshow(removed_bg[...,::-1]);plt.title("removed bg");
plt.subplot(235);plt.imshow(new_bg[...,::-1]);plt.title("background without object");
plt.subplot(236);plt.imshow(final[...,::-1]);plt.title("Final result");
```

در این کد ابتدا عکس را خوانده و آن را به یک عکس HSV تبدیل میکنیم. در مرحله بعد:

از تصویر اصلی، اندازه‌ی عرض و ارتفاع استخراج می‌شود.

تصویر پس‌زمینه با `cv2.resize`به همان اندازه‌ی تصویر اصلی تغییر سایز داده می‌شود، چون باید دقیقا با آن هم‌اندازه باشد تا بتوان به‌درستی آن را جایگزین کرد.

سپس به تعریف محدوده سبز در فضای HSV می رویم:

تعریف محدوده‌ی پایین (low) و بالا (upper) برای رنگ سبز.

این دو آرایه تعیین می‌کنند که کدام پیکسل‌ها در تصویر به عنوان "سبز" در نظر گرفته شوند.

`[35, 50, 50]` → مقدار کمینه برای Hue, Saturation, Value.

`[85, 255, 255]` → مقدار بیشینه برای سبز.

و در نهایت با استفاده از inrange() یک ماسک تعریف می کنیم

- تابع `cv2.inRange`یک ماسک باینری ایجاد می‌کند:
  - پیکسل‌هایی که **در محدوده‌ی سبز تعریف‌شده هستند** → **سفید (255).**
  - بقیه پیکسل‌ها → سیاه (0).
- این ماسک بخش سبز تصویر را جدا می‌کند.

در این کد ما برای تشخیص سوژه ها و اشیا به یک ماسک معکوس هم نیاز داریم وقتی ما می‌خوایم **پرده‌ی سبز رو حذف کنیم** و **سوژه‌ی اصلی (مثلاً یک شخص)** رو نگه داریم، باید بدونیم کدوم بخش از تصویر سبزه و کدوم بخش سوژه‌ست.

اول، چی کار می‌کنیم؟

ما یه **ماسک (mask)** می‌سازیم که فقط **رنگ سبز** رو پیدا می‌کنه.

این ماسک مثل یه عکس سیاه‌وسفیده:

- بخش‌های **سبز** رو به‌صورت **سفید** نشون می‌ده (یعنی: «اینا سبزن!»).
- بقیه بخش‌ها (مثل آدم یا اشیاء) رو **سیاه** نشون می‌ده (یعنی: «اینا سبز نیستن!»).

حالا چرا به ماسک "معکوس" نیاز داریم؟

ما می‌خوایم:

- بخش **سبز** رو حذف کنیم (تا پس‌زمینه رو عوض کنیم)،
- و بخش‌های **غیر سبز** (مثل آدم یا شیء جلوی پرده) رو نگه داریم.

اما ماسک اصلی فقط بخش سبز رو مشخص کرده.

پس برای اینکه بفهمیم **کدوم قسمت غیر سبزه (یعنی همون سوژه)**، باید **رنگ‌های ماسک رو برعکس کنیم**:

- بخش‌های **سفید** (سبزها) تبدیل می‌شن به **سیاه** یعنی حذف کن.
- بخش‌های **سیاه** (غیرسبزها) تبدیل می‌شن به **سفید** یعنی نگه دار.

در مرحله پس زمینه را از تصویر جدید استخراج می کنیم

```
new_bg = cv2.bitwise_and(bg_image, bg_image, mask=mask)
removed_bg = cv2.bitwise_and(input_image, input_image, mask=mask_not_green)

final = cv2.add(removed_bg, new_bg)
```

در این بخش از کد، هدف نهایی ترکیب تصویر سوژه (یعنی تصویر اصلی بدون پرده‌ی سبز) با تصویر پس‌زمینه‌ی جدید است. ابتدا با استفاده از ماسک سبز، از تصویر پس‌زمینه فقط همان قسمت‌هایی که قرار است جایگزین پرده‌ی سبز شوند استخراج می‌شود؛ به

عبارت دیگر، نواحی‌ای از پس‌زمینه‌ی جدید انتخاب می‌شوند که در تصویر اصلی به‌صورت پرده‌ی سبز ظاهر شده بودند. این بخش به‌عنوان "پس‌زمینه جدید" در تصویر نهایی استفاده می‌شود.

سپس با استفاده از ماسک معکوس که نواحی غیرسبز (یعنی سوژه یا جسم جلوی پرده) را مشخص می‌کند، تصویر اصلی از پس‌زمینه‌اش جدا شده و تنها بخش سوژه باقی می‌ماند. در نتیجه، اکنون دو تصویر مجزا در اختیار داریم: یکی تصویر سوژه‌ی بدون پرده سبز، و دیگری تصویر پس‌زمینه‌ی جدید دقیقاً در نواحی‌ای که قبلاً سبز بوده‌اند.

در نهایت، این دو تصویر با استفاده از جمع پیکسلی با هم ترکیب می‌شوند تا تصویر نهایی ساخته شود؛ تصویری که در آن سوژه‌ی اصلی در جلو باقی مانده و پس‌زمینه‌ی جدید جای پرده‌ی سبز را گرفته است، بدون آن‌که این دو بخش با هم تداخل داشته باشند.

در ابتدا، خط `new_bg = cv2.bitwise_and(bg_image, bg_image, mask=mask)` باعث می‌شود که از تصویر پس‌زمینه جدید (یعنی تصویر جایگزین برای پرده‌ی سبز) فقط آن بخش‌هایی باقی بمانند که در تصویر اصلی به‌صورت سبز شناسایی شده‌اند. این کار با استفاده از عملگر منطقی AND انجام می‌شود؛ به این صورت که تصویر پس‌زمینه جدید با خودش AND می‌شود، اما فقط در نواحی‌ای که ماسک مقدار ۲۵۵ (سفید) دارد. در نتیجه، نواحی متناظر با پرده سبز در تصویر پس‌زمینه انتخاب و فعال می‌شوند و باقی بخش‌ها صفر یا سیاه باقی می‌مانند.

سپس، در خط `removed_bg = cv2.bitwise_and(input_image, input_image, mask=mask_not_green)`، دقیقاً همین منطق روی تصویر اصلی اعمال می‌شود؛ با این تفاوت که این‌بار از ماسک معکوس استفاده می‌شود. این ماسک بخش‌هایی از تصویر را که سبز نیستند (یعنی سوژه اصلی مانند انسان یا هر جسم جلوی پرده) مشخص می‌کند. بنابراین، این خط باعث می‌شود فقط سوژه از تصویر اصلی باقی بماند و پرده‌ی سبز حذف شود.

در نهایت، خط `final = cv2.add(removed_bg, new_bg)` دو تصویر تولیدشده از مراحل قبلی را با هم ترکیب می‌کند. چون این دو تصویر از نظر مکانی مکمل یکدیگر هستند (یعنی سوژه در یکی وجود دارد و پس‌زمینه در دیگری)، جمع پیکسلی آن‌ها تصویر نهایی را می‌سازد؛ تصویری که در آن سوژه‌ی اصلی به‌درستی حفظ شده و پس‌زمینه‌ی جدید جایگزین پرده‌ی سبز شده است، بدون تداخل بین اجزا.

حذف پرده سبز با استفاده از فضای رنگی LAB :

با استفاده از این کد می توانیم پرده سبز را حذف کنیم

```
# Load video clip
cap = cv2.VideoCapture('./videos/greenscreen.mp4')
bg_image = cv2.imread("images/balloon.png")

# Get the height and width of the frame (required to be an interger)
w = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
h = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
#resize to frame size
bg_image = cv2.resize(bg_image,(w, h))

while True:
    ret, frame = cap.read()
    if not ret:
        break
    background = bg_image.copy()
    lab = cv2.cvtColor(frame, cv2.COLOR_BGR2LAB)
    a_channel = lab[:,:,1]
    _, th = cv2.threshold(a_channel,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
    mask_green = cv2.bitwise_not(th)
    removed_bg = cv2.bitwise_and(frame, frame, mask=th)
    background = cv2.bitwise_and(background, background, mask=mask_green)

    final = cv2.add(removed_bg, background)
    cv2.imshow("result", final)
    if cv2.waitKey(15) & 0xFF == ord('q'):
        break


cap.release()
cv2.destroyAllWindows()
plt.imshow(cv2.cvtColor(final, cv2.COLOR_BGR2RGB)); plt.axis('off'); plt.show();
```

در کد فوق پس از خواند عکس با استفاده از این کد:

```
w = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
h = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
```

عرض (w) و ارتفاع (h) فریم‌های ویدیو رو از خود ویدیو می‌خوانند. این ابعاد به صورت عدد صحیح ذخیره می‌شوند.

در مرحله بعد تصویر پس‌زمینه با اندازه‌ی فریم‌های ویدیو تغییر اندازه داده می‌شود تا سایزش با فریم‌ها هماهنگ شود.

در این مرحله:

```
channel = lab[:,:,1]
```

به این معنی است که از تصویر `lab` که در فضای رنگ LAB ذخیره شده، کانال دوم (index=1) را جدا می‌کنیم.

فضای رنگ LAB سه کانال دارد:

- **L**: روشنایی — (Lightness) مقدار روشن یا تاریک بودن پیکسل
- **a**: محور رنگ که نشان‌دهنده مقدار بین رنگ سبز تا قرمز است
- **b**: محور رنگ که نشان‌دهنده مقدار بین رنگ آبی تا زرد است

کانال `a` در واقع میزان "گرایش رنگ" را در جهت سبز-قرمز اندازه‌گیری می‌کند. یعنی:

- اگر مقدار کانال `a` برای یک پیکسل منفی یا پایین باشد، آن پیکسل گرایش به رنگ سبز دارد.
- اگر مقدار کانال `a` بالا یا مثبت باشد، آن پیکسل گرایش به رنگ قرمز دارد.

در این کد، چون هدف حذف پرده سبز است، کانال `a` کمک می‌کند که پیکسل‌های سبز بهتر از بقیه رنگ‌ها تفکیک شوند.

بنابراین `a_channel = lab[:,:,1]` یعنی استخراج فقط همان کانال رنگی که در آن تمایز رنگ سبز و قرمز به وضوح دیده می‌شود تا بتوان با اعمال آستانه‌گذاری (thresholding) پرده سبز را تشخیص داد.

خلاصه: این خط فقط کانال رنگی "a" (سبز-قرمز) را از تصویر LAB جدا می‌کند تا بتوان پرده سبز را بهتر تشخیص داد.