به نام خدا

پردازش تصویر و بینایی ماشین

آموزش عملی پردازش تصویر:
نکات کلیدی+کدهای پایتون

مولف :

علیرضا شایگان

انتشارات ارسطو

(سازمان چاپ و نشر ایران – ۱۴۰۴)

سرشناسه : شایگان، علیرضا، ۱۳۶۰-
عنوان و نام پدیدآور : پردازش تصویر و بینایی ماشین[کتاب]: آموزش عملی پردازش تصویر: نکات کلیدی+کدهای پایتون/ علیرضا شایگان.
مشخصات نشر : انتشارات ارسطو (سازمان چاپ و نشر ایران)،۱۴۰۴.
مشخصات ظاهری : ۶۱۵ص.: مصور.
شابک : ۴-۲۴۶-۴۵۵-۶۲۲-۹۷۸
وضعیت فهرست نویسی : فیپا
عنوان دیگر : آموزش عملی پردازش تصویر: نکات کلیدی+کدهای پایتون.
موضوع : عکس‌پردازی -- روش‌های رقمی
موضوع : Image processing -- Digital techniques
موضوع : بینایی ماشین -- برنامه‌نویسی
موضوع : Computer vision -- Programming
موضوع : پایتون (زبان برنامه‌نویسی کامپیوتر)
موضوع : Python (Computer program language)
رده بندی کنگره : TA۱۶۳۲
رده بندی دیویی : ۶۲۱/۳۶۷
شماره کتابشناسی ملی : ۱۰۰۵۱۵۴۵

نام کتاب : پردازش تصویر و بینایی ماشین
(آموزش عملی پردازش تصویر: نکات کلیدی+کدهای پایتون)
مولف : علیرضا شایگان
ناشر : انتشارات ارسطو (سازمان چاپ و نشر ایران)
صفحه آرایی، تنظیم و طرح جلد: پروانه مهاجر
تیراژ : ۱۰۰۰ جلد
نوبت چاپ : اول – ۱۴۰۴
چاپ : زبرجد
قیمت : ۶۱۵۰۰۰ تومان
فروش نسخه الکترونیکی - کتاب‌رسان :
https://chaponashr.ir/ketabresan
شابک : ۴-۲۴۶-۴۵۵-۶۲۲-۹۷۸
تلفن مرکز پخش : ۰۹۱۲۰۲۳۹۲۵۵
www. chaponashr. ir

انتشارات ارسطو
چاپ و نشر ایران
Chaponashr.ir

پیشگفتار

در عصر حاضر، جهان در حال تجربه تحولات شگرفی در زمینه فناوری‌های دیجیتال و هوش مصنوعی است. یکی از حوزه‌های برجسته و پیشرو در این انقلاب، پردازش تصویر و بینایی ماشین است که به انسان‌ها امکان می‌دهد تا کامپیوترها را برای درک و تحلیل دنیای بصری پیرامونشان به‌کار گیرند. این فناوری‌ها در بسیاری از جنبه‌های زندگی روزمره، از تشخیص چهره در دستگاه‌های هوشمند تا رباتیک پیشرفته، خودروهای خودران، پزشکی و حتی هنر، نقش بسزایی ایفا می‌کنند.

هدف از نگارش این کتاب، ارائه مجموعه‌ای جامع از مفاهیم، تکنیک‌ها و ابزارهای مرتبط با پردازش تصویر و بینایی ماشین است. این اثر نه‌تنها برای علاقه‌مندان و پژوهشگران مبتدی طراحی شده است، بلکه به متخصصانی که به دنبال گسترش دانش و مهارت‌های خود در این حوزه هستند نیز کمک می‌کند. ساختار کتاب به گونه‌ای است که خواننده را گام‌به‌گام از مبانی اولیه به موضوعات پیشرفته هدایت می‌کند و در هر مرحله با مثال‌های کاربردی و پروژه‌های عملی، یادگیری را تسهیل می‌کند.

این کتاب شامل مباحث گوناگونی از جمله معرفی پردازش تصویر و بینایی ماشین، آشنایی با کتابخانه‌ها و ابزارهای مرتبط، تحلیل و بهبود کیفیت تصاویر، کار با فیلترهای تصویر، تشخیص اشیاء و ویژگی‌های کلیدی، ردیابی در ویدئوها، و استفاده از یادگیری ماشین برای تحلیل تصاویر است. تنوع موضوعات و گستره پوشش این کتاب، آن را به مرجعی ارزشمند برای یادگیری و استفاده عملی تبدیل کرده است.

امید است این کتاب بتواند به‌عنوان پلی میان دانش نظری و کاربرد عملی، زمینه‌ساز توسعه پروژه‌های خلاقانه و پیشرفت‌های علمی در حوزه پردازش تصویر و بینایی ماشین باشد. از تمامی خوانندگان عزیز دعوت می‌شود تا نظرات و تجربیات خود را با ما به اشتراک بگذارند و در مسیر تکامل این دانش سهیم شوند.

علیرضا شایگان

فهرست مطالب

فصل اول

پردازش تصویر و بینایی ماشین – مقدمه

۱- فصل اول: پردازش تصویر و بینایی ماشین - مقدمه

پردازش تصویر و بینایی ماشین هر دو به تحلیل و تفسیر داده‌های تصویری مربوط می‌شوند، اما تفاوت‌هایی نیز دارند :

۱- ۱- پردازش تصویر (Image Processing)

- تعریف : پردازش تصویر به تکنیک‌ها و الگوریتم‌هایی اشاره دارد که بر روی تصاویر دیجیتال اعمال می‌شود تا اطلاعاتی از تصویر استخراج شود یا کیفیت تصویر بهبود یابد.

- هدف: هدف اصلی پردازش تصویر معمولاً بهبود کیفیت تصویر، فیلتر کردن نویز، شفاف‌سازی و یا استخراج ویژگی‌های خاص از تصویر است.

- کاربردها : شامل کاربردهایی مانند فیلتر کردن تصاویر، تشخیص لبه‌ها، فشرده‌سازی تصویر و تصحیح رنگ

۱- ۲- بینایی ماشین (Computer Vision)

- تعریف : بینایی ماشین به سیستمی اشاره دارد که قادر است از طریق پردازش تصویر و الگوریتم‌های پیچیده، درک و تحلیل معنایی از تصاویر یا ویدیوها انجام دهد.

- هدف: هدف بینایی ماشین این است که به سیستم‌ها اجازه دهد تا تصاویر و ویدیوها را مانند انسان‌ها درک کنند و تصمیم‌گیری کنند.

– کاربردها : شامل شناسایی اشیاء، تشخیص چهره، تحلیل حرکت، ویدیو آنالیز و استنباط معنایی از تصاویر.

به طور کلی، پردازش تصویر بیشتر بر روی تکنیک‌های پایه‌ای و بهبود کیفیت تصویر تمرکز دارد، در حالی که بینایی ماشین به دنبال درک و تحلیل معنایی تصاویر و ویدیوها به صورت پیچیده‌تر و عمیق‌تر است.

۱– ۳– منشاء ایده‌های پردازش تصویر و بینایی ماشین

ایده‌های پردازش تصویر و بینایی ماشین از تلاش‌ها برای شبیه‌سازی و بهبود توانایی‌های بصری انسان‌ها در کامپیوترها و سیستم‌های هوش مصنوعی نشأت گرفته است. برخی از منابع و پیشینه‌های اصلی این ایده‌ها عبارتند از :

۱. مطالعه سیستم‌های بینایی زیستی : محققان از سیستم‌های بینایی در طبیعت، به ویژه چشم و مغز انسان، الهام گرفته‌اند. تلاش‌ها برای درک چگونگی پردازش و تحلیل تصاویر توسط سیستم‌های زیستی، به توسعه الگوریتم‌ها و تکنیک‌های پردازش تصویر و بینایی ماشین کمک کرده است.

۲. پیشرفت‌های ریاضی و محاسباتی : توسعه روش‌های ریاضی و محاسباتی، از جمله تحلیل فوریه، تبدیل‌های ماتریسی و فیلترهای دیجیتال، به پردازش و تحلیل تصاویر دیجیتال کمک کرده است.

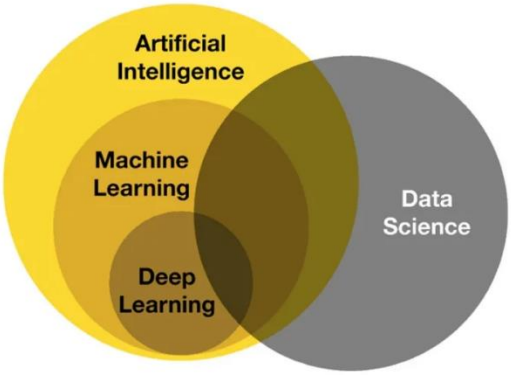
۳. تحقیقات علمی و دانشگاهی : تحقیقات اولیه در زمینه بینایی ماشین و پردازش تصویر در دانشگاه‌ها و مراکز تحقیقاتی علمی آغاز شد. این تحقیقات شامل کار بر روی الگوریتم‌های شناسایی الگو، پردازش تصویر دیجیتال و تحلیل تصاویر بود.

۴. پیشرفت‌های فناوری و سخت‌افزار : با پیشرفت‌های فناوری، از جمله توسعه دوربین‌های دیجیتال، سنسورها و قدرت پردازش کامپیوترها، امکان پردازش و تحلیل داده‌های تصویری به صورت سریع و دقیق فراهم شد.

۵. پروژه‌های کاربردی و صنعتی : نیازهای عملی و صنعتی، مانند سیستم‌های نظارت تصویری، شناسایی خودروها و تشخیص چهره، به توسعه تکنیک‌های پردازش تصویر و بینایی ماشین کمک کرد. این پروژه‌ها نیاز به الگوریتم‌های دقیق و کارآمد برای تحلیل داده‌های تصویری را به وجود آوردند.

به طور کلی، ترکیبی از مطالعات علمی، پیشرفت‌های فناوری و نیازهای عملی به شکل‌گیری و توسعه پردازش تصویر و بینایی ماشین منجر شد.

۱- ۴- نمودار رابطه بینایی ماشین (علم داده) با هوش مصنوعی



شکل ۱- نمودار رابطه بینایی ماشین (علم داده) با هوش مصنوعی

۱- ۵- کتابخانه‌ها و ابزارهای مختلف در پردازش تصویر و بینایی ماشین

در پردازش تصویر و بینایی ماشین، کتابخانه‌ها و ابزارهای مختلفی وجود دارند که می‌توانند در پیاده‌سازی الگوریتم‌ها و مدل‌های پیچیده کمک کنند. در اینجا به معرفی برخی از کتابخانه‌های رایج در این حوزه‌ها پرداخته می‌شود.

۱- ۵- ۱-کتابخانه‌های پردازش تصویر

1- OpenCV

- یکی از قدرتمندترین و پرکاربردترین کتابخانه‌ها برای پردازش تصویر و بینایی ماشین. OpenCV ابزارها و الگوریتم‌های مختلفی برای عملیات‌های پایه‌ای و پیچیده تصویر فراهم می‌کند.

- زبان‌ها : پایتون و C++ و Java

2- Pillow **(**PIL**)**

- کتابخانه‌ای ساده و راحت برای پردازش تصویر در . Pillow برای کارهای ابتدایی مانند بارگذاری، ذخیره و اصلاح تصاویر مناسب است.
- زبان‌ها : پایتون

3- scikit-image

- یک کتابخانه قدرتمند برای پردازش تصویر که به عنوان بخشی از مجموعه scikit-learn توسعه داده شده است. این کتابخانه شامل ابزارهایی برای تحلیل تصویر و الگوریتم‌های پردازش تصویر است.
- زبان‌ها : پایتون

۱- ۵- ۲-کتابخانه‌های بینایی ماشین

1- OpenCV

- علاوه بر پردازش تصویر، OpenCV ابزارهای پیشرفته‌ای برای بینایی ماشین، از جمله شناسایی اشیاء، تشخیص چهره و تحلیل ویدیو فراهم می‌کند.
- زبان‌ها : پایتون و C++ و Java

2- dlib

- توضیحات کتابخانه‌ای قدرتمند برای بینایی ماشین و یادگیری ماشین که شامل الگوریتم‌های شناسایی چهره، شبیه‌سازی مدل‌های بینایی و ویژگی‌های دیگر است.
- زبان‌ها : پایتون و C++

3- SimpleCV

- توضیحات یک فریم‌ورک ساده و کاربرپسند برای بینایی ماشین در که به تسهیل فرآیند توسعه پروژه‌های بینایی ماشین کمک می‌کند.
- زبان‌ها : پایتون

۱- ۵- ۳-کتابخانه‌های یادگیری عمیق برای بینایی ماشین

1- TensorFlow

- یک فریم‌ورک متن باز برای یادگیری ماشین و یادگیری عمیق که توسط Google توسعه داده شده است. TensorFlow ابزارهای پیشرفته‌ای برای ساخت و آموزش مدل‌های بینایی ماشین فراهم می‌کند.
- زبان‌ها : پایتون و C++ و Java

2- Keras

- یک API سطح بالا برای یادگیری عمیق که می‌تواند بر روی TensorFlow و دیگر فریم‌ورک‌ها اجرا شود. Keras طراحی ساده‌ای دارد و برای توسعه سریع مدل‌های یادگیری عمیق مناسب است.
- زبان‌ها : پایتون

3- PyTorch

- فریم‌ورکی برای یادگیری عمیق که توسط Facebook توسعه داده شده است. PyTorch به خاطر قابلیت‌های داینامیک و تسهیل در پیاده‌سازی مدل‌های پیچیده، محبوبیت زیادی دارد.
- زبان‌ها : پایتون و C++

4- Fastai

- یک کتابخانه یادگیری عمیق که بر روی PyTorch ساخته شده است و به منظور ساده‌سازی فرآیند آموزش و توسعه مدل‌های یادگیری عمیق طراحی شده است.
- زبان‌ها : پایتون

این کتابخانه‌ها ابزارهای قدرتمندی هستند که در پردازش تصاویر، توسعه سیستم‌های بینایی ماشین و پیاده‌سازی مدل‌های یادگیری عمیق کمک خواهند کرد.

۱- ۶- شروع پردازش تصویر و بینایی ماشین

۱- ۶- ۱-استفاده از محیط Jupyter Notebook برای یادگیری پردازش تصویر

استفاده از Jupyter Notebook برای پردازش تصویر مزایای زیادی دارد که آن را به یک ابزار محبوب برای این کار تبدیل کرده است. در اینجا به دلایل کلیدی برای استفاده از Jupyter Notebook در پردازش تصویر اشاره می‌شود

۱. توسعه تعاملی و مرحله به مرحله

- در Jupyter Notebook، می‌توان کد را به صورت مرحله به مرحله اجرا کرد و در هر مرحله نتایج را به صورت فوری مشاهده نمود. این قابلیت، این امکان را می‌دهد که هر تغییر در کد را سریعاً بررسی و ارزیابی کرد. در پردازش تصویر، مشاهده مستقیم خروجی پردازش (مانند فیلترها، تشخیص لبه‌ها، و غیره) می‌تواند بسیار مفید باشد.
- مزیت : سریع‌تر، اشکال‌زدایی و اصلاح کدها.

۲. نمایش گرافیکی و تجسم بصری

- Jupyter Notebook قابلیت ادغام ساده با ابزارهای تجسم داده مانند Matplotlib و OpenCV را دارد. می‌توان تصاویر و نتایج پردازش تصویر را مستقیماً درون

نوت‌بوک نمایش داد، بدون نیاز به ذخیره کردن و باز کردن مجدد فایل‌ها در ابزارهای دیگر.

- مزیت : تجسم و تحلیل فوری تصاویر و داده‌های بصری.

۳. محیط آموزشی و تحقیقاتی مناسب

- Jupyter Notebook یک محیط عالی برای یادگیری و تحقیق است. به‌خاطر طبیعت تعاملی آن، بسیاری از دانشجویان و محققان از آن برای یادگیری پردازش تصویر و انجام پروژه‌های تحقیقاتی استفاده می‌کنند. می‌توان تکه‌های کد را نوشته و آن‌ها را آزمایش کرده و در صورت نیاز توضیحات متنی به همراه تصاویر اضافه نمود.
- مزیت : تسهیل یادگیری و مستندسازی پروژه‌ها.

۴. مستندسازی همزمان با کد

- می‌توان همراه با کدها، یادداشت‌ها، توضیحات، فرمول‌ها، و تفسیرهای متنی را در نوت‌بوک قرار داد. این ویژگی برای پروژه‌های تحقیقاتی یا آموزشی که نیاز به توضیح هر مرحله دارند، بسیار مفید است.
- مزیت : ایجاد مستندات جامع همراه با کد که درک فرآیندهای پردازش تصویر را آسان‌تر می‌کند.

۵. پشتیبانی از کتابخانه‌های قدرتمند

- Jupyter Notebook از کتابخانه‌های مختلف پایتون مانند OpenCV، scikit-image، Pillow و TensorFlow به خوبی پشتیبانی می‌کند. این کتابخانه‌ها ابزارهای قدرتمندی برای پردازش تصویر و بینایی ماشین فراهم می‌کنند.
- مزیت : استفاده راحت و یکپارچه از طیف گسترده‌ای از ابزارهای پردازش تصویر.

۶. قابلیت به اشتراک‌گذاری آسان

- فایل‌های Jupyter Notebook با پسوند ipynb. ذخیره می‌شوند و می‌توانند به راحتی در بین همکاران به اشتراک گذاشته شوند. همچنین امکان استفاده از پلتفرم‌هایی مانند Google Colab برای اجرای نوت‌بوک‌ها در محیط‌های ابری وجود دارد.
- مزیت : تسهیل همکاری تیمی و به اشتراک‌گذاری کد و نتایج پردازش تصویر.

۷. توسعه مدل‌های یادگیری ماشین و یادگیری عمیق

- بسیاری از پروژه‌های پردازش تصویر نیاز به استفاده از تکنیک‌های یادگیری ماشین و یادگیری عمیق دارند. Jupyter Notebook به دلیل سازگاری بالا با کتابخانه‌هایی مانند TensorFlow و PyTorch و قابلیت آموزش و تست مدل‌ها در همان محیط، انتخابی عالی برای توسعه مدل‌های بینایی ماشین است.
- مزیت : اجرای مدل‌های پیچیده یادگیری ماشین و بینایی ماشین و پردازش تصویر.

Jupyter Notebook به دلیل قابلیت‌های تعاملی، تجسم بصری و پشتیبانی از کتابخانه‌های پیشرفته، به محیطی بسیار مناسب برای پردازش تصویر تبدیل شده است. این محیط این امکان را می‌دهد تا کدها را به راحتی اجرا کرده، نتایج را مشاهده نموده و فرآیندهای مختلف به طور موثر بررسی و تحلیل گردد.

۱- ۶- ۲-خواندن تصویر در محیط ژوپیتر نوت بوک

برای خواندن و نمایش تصویر در محیط Jupyter Notebook، می‌توان از کتابخانه‌های مختلفی مانند OpenCV، Pillow (PIL)، و Matplotlib استفاده نمود. در اینجا چند روش رایج برای خواندن و نمایش تصویر آورده شده است:

۱. استفاده از Pillow (PIL)

```
from PIL import Image
import matplotlib.pyplot as plt
                                                    خواندن تصویر

image = Image.open('path_to_your_image.jpg')
                                                    نمایش تصویر

plt. imshow(image)
                                                    حذف محورها

plt. axis('off')
plt.show()
```

۲. استفاده از OpenCV

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

خواندن تصویر با OpenCV

```
image = cv2.imread('path_to_your_image.jpg')
```

OpenCV تصاویر را در فضای رنگی BGR باز می‌کند، برای تبدیل به RGB :

```
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

نمایش تصویر

```
plt.imshow(image_rgb)
```

حذف محورها

```
plt.axis('off')
plt.show()
```

۳. استفاده از Matplotlib (به‌تنهایی)

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

خواندن تصویر

```
image = mpimg.imread('path_to_your_image.jpg')
```

نمایش تصویر

```
plt.imshow(image)
```

حذف محورها

```
plt.axis('off')
plt.show()
```

نکات

- به جای 'path_to_your_image.jpg' باید مسیر فایل تصویری را وارد نمود.
- کتابخانه Pillow برای کارهای ساده مانند باز کردن و نمایش تصاویر بسیار کاربرپسند است.
- OpenCV قابلیت‌های بیشتری برای پردازش تصویر و تحلیل‌های پیچیده‌تر ارائه می‌دهد.

۱- ۶- ۳-استخراج ویژگی‌های اصلی یک تصویر

استخراج اطلاعات مختلف درباره یک تصویر مانند اندازه (ابعاد)، نوع (فرمت داده‌ها) و سایر ویژگی‌های مرتبط برای استخراج اطلاعات مختلف درباره یک تصویر مانند اندازه (ابعاد)، نوع (فرمت داده‌ها) و سایر ویژگی‌های مرتبط، می‌توان از OpenCV و توابع مرتبط در این کتابخانه استفاده نمود. در اینجا نحوه استخراج اطلاعاتی مانند اندازه تصویر، نوع داده‌ها و تعداد کانال‌ها آورده شده است:

۱. اندازه تصویر (ابعاد)

- برای بدست آوردن اندازه تصویر (عرض و ارتفاع)، از خاصیت shape استفاده می‌شود.

۲. نوع تصویر (نوع داده‌ها)

- نوع داده‌های پیکسل‌های تصویر را می‌توان با استفاده از تابع dtype بدست آورد.

۳. تعداد کانال‌ها

- تصویر می‌تواند یک تصویر رنگی (با ۳ کانال) یا خاکستری (با ۱ کانال) باشد. با استفاده از خاصیت shape می‌توان تعداد کانال‌ها را نیز بدست آورد.

کد نمونه برای استخراج ویژگی‌های اصلی یک تصویر :

```
import cv2

خواندن تصویر

image = cv2.imread('path_to_image.jpg')

بدست آوردن ابعاد تصویر (ارتفاع، عرض و تعداد کانال‌ها)

height, width, channels = image.shape

بدست آوردن نوع داده‌های پیکسل‌ها

image_type = image.dtype

چاپ اطلاعات تصویر

print(f'ابعاد تصویر : {width}x{height}')

print(f'تعداد کانال‌ها : {channels}')

print(f'نوع داده‌های تصویر : {image_type}')
```

توضیحات

۱. ابعاد تصویر : ویژگی shape تصویر را برمی‌گرداند که به ترتیب شامل (height, width, channels) است.

- height (ارتفاع) : تعداد پیکسل‌های عمودی تصویر.
- width (عرض) : تعداد پیکسل‌های افقی تصویر.
- channels : تعداد کانال‌های تصویر. برای تصاویر رنگی (RGB) معمولاً ۳ و برای تصاویر خاکستری ۱ است.

۲. نوع داده‌ها : نوع داده‌های پیکسل‌ها (مثلاً uint8 برای مقادیر ۰ تا ۲۵۵) که نشان‌دهنده نوع داده مورد استفاده برای هر پیکسل است.

این اطلاعات می‌تواند برای پردازش‌های بعدی مانند تغییر اندازه تصویر، فیلترگذاری، یا تحلیل‌های پیچیده‌تر در پردازش تصویر مفید باشد.

۱- ۶- ۴-ذخیره یک تصویر در یک مسیر مشخص

برای ذخیره یک تصویر در یک مسیر مشخص با استفاده از OpenCV، می‌توان از تابع cv2.imwrite () استفاده نمود. این تابع این امکان را می‌دهد تصویر پردازش‌شده یا اصلاح‌شده را در قالب فایل (مانند JPG یا PNG) ذخیره نمود.

کد نمونه برای ذخیره تصویر

```
import cv2
```

```
خواندن تصویر
image = cv2.imread('path_to_image.jpg')
پردازش تصویر (مثلاً تبدیل به تصویر خاکستری)
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
مسیر ذخیره‌سازی تصویر
save_path = 'path_to_save_image/gray_image.jpg'
ذخیره تصویر در مسیر مشخص
cv2.imwrite(save_path, gray_image)
print(f"تصویر با موفقیت در {save_path} ذخیره شد")
```

توضیحات

۱. cv2.imread() : تصویر را از مسیر مشخص می‌خواند.

۲. cv2.cvtColor() : برای تبدیل تصویر به فضای رنگی مورد نظر (در اینجا خاکستری).

۳. cv2.imwrite() : تصویر را در مسیر مشخص ذخیره می‌کند. تابع دو پارامتر می‌گیرد :

- پارامتر اول : مسیر ذخیره (به همراه نام و فرمت فایل، مانند jpg یا png).
- پارامتر دوم : تصویر که باید ذخیره شود (تصویر پردازش‌شده).

نکات

- فرمت تصویر در هنگام ذخیره‌سازی با توجه به پسوند فایل تعیین می‌شود. برای مثال، اگر پسوند jpg باشد، تصویر به فرمت JPG ذخیره می‌شود.

- مطمئن شوید که مسیر ذخیره‌سازی صحیح است و دسترسی لازم برای نوشتن در آن دارید.

با این روش، می‌توان تصاویر پردازش‌شده خود را به راحتی در مسیرهای دلخواه ذخیره نمود.

۱- ۶- ۵-نمایش صحیح کانالها در تصاویر رنگی و سیاه سفید

در پردازش تصویر با OpenCV و کتابخانه‌های مانند Matplotlib، تغییر فضاهای رنگی (مانند تبدیل تصویر رنگی از BGR به RGB) و تنظیم نحوه نمایش تصاویر (مانند cmap برای تصاویر خاکستری) دو موضوع بسیار رایج هستند. در اینجا به جزئیات هر کدام از این موارد پرداخته می‌شود.

۱. تبدیل فضای رنگی از BGR به RGB

OpenCV به طور پیش‌فرض تصاویر رنگی را در فضای BGR (آبی، سبز، قرمز) بارگذاری می‌کند، در حالی که اکثر کتابخانه‌های نمایش تصویر مانند Matplotlib از فضای رنگی RGB استفاده می‌کنند. برای نمایش صحیح یک تصویر رنگی بارگذاری شده توسط OpenCV، باید ترتیب کانال‌ها را از BGR به RGB تغییر داد.

تبدیل از BGR به RGB

- در OpenCV، می‌توان از کد cv2.COLOR_BGR2RGB برای تغییر فضای رنگی از BGR به RGB استفاده نمود.

کد نمونه

```
import cv2
import matplotlib.pyplot as plt
خواندن تصویر با OpenCV (فضای رنگی BGR)
image_bgr = cv2.imread('path_to_image.jpg')
تبدیل فضای رنگی از BGR به RGB
image_rgb = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2RGB)
نمایش تصویر در فضای RGB با Matplotlib
plt.imshow(image_rgb)
حذف محورها
plt.axis('off')
plt.show()
```

استفاده از کد۴

- کد ۴ : گاهی اوقات در متدهای قدیمی برای تغییر فضای رنگی از BGR به RGB به عنوان مقدار ثابت از ۴ استفاده می‌شد. اما استفاده از cv2.COLOR_BGR2RGB بسیار رایج‌تر و خواناتر است.

برعکس کردن کانال‌های رنگی

برای برعکس کردن کانال‌های رنگی (مانند تبدیل BGR به RGB یا برعکس)، یک روش ساده و سریع استفاده از برش آرایه‌ها در پایتون با استفاده از [1– : : , ...] است. این روش تمام کانال‌های تصویر را در جهت معکوس مرتب می‌کند.

نحوه استفاده از [1– , ...] برای برعکس کردن کانال‌های رنگی

اگر فرضا تصویری دارید که به صورت پیش‌فرض در OpenCV با فرمت BGR بارگذاری شده است. با استفاده از [1– : : , ...] می‌توان کانال‌ها را به صورت معکوس تبدیل نمود و آن را به RGB تغییر داد.

```
import cv2
import matplotlib.pyplot as plt
خواندن تصویر با OpenCV (فضای رنگی BGR)
image_bgr = cv2.imread('path_to_image.jpg')
برعکس کردن کانال‌ها (تبدیل BGR به RGB)
image_rgb = image_bgr[…, : : -1]
نمایش تصویر در فضای RGB با Matplotlib
```

```
plt.imshow(image_rgb)
plt.axis('off')
```

حذف محورها

```
plt.show()
```

توضیح [... , 1-]

به معنی استفاده از همه ابعاد قبلی است (همه سطرها و همه ستون‌ها).

[: : -1] به معنی معکوس کردن ترتیب عناصر در آخرین بعد (در اینجا معکوس کردن کانال‌های رنگی).

چرا از این روش استفاده می‌شود؟

این روش سریع و ساده است و برای بسیاری از کاربردهای ساده کافی است. با استفاده از این روش، می‌توان به راحتی بدون استفاده از توابع اضافی، ترتیب کانال‌های رنگی را تغییر داد.

مقایسه با cv2.cvtColor()

هر دو روش (cv2.COLOR_BGR2RGB و [... , : : -1]) برای تبدیل BGR به RGB یا برعکس استفاده می‌شوند. روش استفاده از [... , : : -1] سریع‌تر و کوتاه‌تر است، اما ممکن است در شرایطی که نیاز به تغییرات پیچیده‌تر در فضای رنگی هست، استفاده از cv2.cvtColor() مفیدتر باشد.

برای تغییر سریع و ساده ترتیب کانال‌های رنگی (مثلاً تبدیل BGR به RGB یا برعکس)، استفاده از [: : -1 , ...] بسیار کارآمد است. برای تبدیل‌های پیچیده‌تر در فضای رنگی، از ()cv2.cvtColor استفاده نمود.

۲. استفاده از cmap برای نمایش تصاویر خاکستری

برای تصاویر خاکستری، OpenCV به صورت پیش‌فرض تصاویر را به صورت تک کاناله می‌خواند. زمانی که خواسته شود این تصاویر را با Matplotlib نمایش داد، می‌توان از پارامتر cmap='gray' برای نمایش صحیح آنها استفاده نمود.

کد نمونه برای نمایش تصویر خاکستری با cmap='gray'

```
import cv2
import matplotlib.pyplot as plt
خواندن تصویر به صورت خاکستری
gray_image = cv2.imread('path_to_image.jpg', cv2.IMREAD_GRAYSCALE)
نمایش تصویر خاکستری با استفاده از cmap='gray'
plt.imshow(gray_image, cmap='gray')
حذف محورها
plt.axis('off')
plt.show()
```

- cmap='gray' : در Matplotlib، cmap یا "color map" یک پارامتر است که این امکان را می‌دهد نقشه رنگی برای نمایش تصویر تنظیم نمود. برای تصاویر

خاکستری، باید cmap='gray' باشد تا تصویر به درستی به صورت سیاه و سفید نمایش داده شود.

- برای تصاویر رنگی : از cv2.cvtColor() با cv2.COLOR_BGR2RGB استفاده نمود تا تصاویر BGR که توسط OpenCV بارگذاری شده‌اند را به RGB تبدیل نمود و سپس با Matplotlib نمایش داد.
- برای تصاویر خاکستری : از cmap='gray' استفاده نمود تا اطمینان حاصل شود تصویر خاکستری به درستی نمایش داده می‌شود. این روش‌ها امکان می‌دهند تصاویر رنگی و خاکستری را به درستی و به شکل مورد نظر پردازش و نمایش داد.

۱- ۶- ۶-برش زدن قسمتی از تصویر

برای برش تصویر در پایتون با استفاده از کتابخانه‌های مختلف می‌توان دو رویکرد عمده داشت :

۱. برش زدن تصویر با استفاده از دستور clip (کتابخانه OpenCV)

کتابخانه OpenCV یکی از قدرتمندترین کتابخانه‌ها برای پردازش تصویر است. از این کتابخانه می‌توان برای برش زدن تصاویر استفاده کرد. برای برش می‌توان از محدوده (Region of Interest) به صورت مستقیم بهره برد. مثال زیر نحوه برش با استفاده از OpenCV را نشان می‌دهد.

```
import cv2
```

```
بارگذاری تصویر
image = cv2.imread('image.jpg')
تعریف محدوده برش : [y1 : y2, x1 : x2]
cropped_image = image[300:100 , 200:50]
ذخیره تصویر برش‌خورده
cv2.imwrite('cropped_image.jpg', cropped_image)
```

۲. برش نسبتی از تصویر (کتابخانه Pillow)

کتابخانه Pillow که نسخه پیشرفته‌تر PIL است، برای کار با تصاویر و پردازش آنها به شکل نسبتی می‌تواند استفاده شود. می‌توان به کمک توابع موجود در این کتابخانه، بخشی از تصویر را بر اساس درصدی از ابعاد تصویر اصلی برش داد.

```
from PIL import Image
بارگذاری تصویر
image = Image.open('image.jpg')
دریافت ابعاد تصویر
width, height = image.size
محاسبه محدوده برش به صورت نسبتی (مثلاً ۲۵ درصد از تصویر)
left = width 0.25
top = height 0.25
right = width 0.75
bottom = height 0.75
برش تصویر
cropped_image = image.crop((left, top, right, bottom))
```

```
ذخیره تصویر برش‌خورده
cropped_image. save('cropped_image.jpg')
```

در روش اول، مستقیماً از مختصات پیکسل‌ها برای برش استفاده نمود، اما در روش دوم با استفاده از درصدی از ابعاد تصویر می‌توان برش انجام داد.

۱- ۶- ۷-تجزیه و ترکیب کانال‌های رنگی تصویر با دستورات split و merge

دستورات split و merge در پردازش تصویر برای تجزیه و ترکیب کانال‌های رنگی تصویر استفاده می‌شوند. معمولاً تصاویر رنگی دارای سه کانال RGB (قرمز، سبز، آبی) هستند که هر کانال نشان‌دهنده شدت یک رنگ است. با استفاده از این دستورات می‌توان کانال‌های تصویر را جدا کرده و سپس آن‌ها را مجدداً با هم ترکیب نمود.

۱. دستور split (جدا کردن کانال‌های تصویر)

دستور split کانال‌های مختلف یک تصویر را جدا می‌کند. به عنوان مثال، یک تصویر RGB را به سه کانال قرمز، سبز و آبی تقسیم می‌کند.

```
import cv2
بارگذاری تصویر
image = cv2.imread('image.jpg')
جدا کردن کانال‌های رنگی
b, g, r = cv2.split(image)
نمایش کانال‌های جدا شده
cv2.imshow('Blue Channel', b)
cv2.imshow('Green Channel', g)
```

```
cv2.imshow('Red Channel', r)
cv2.waitKey(۰)

cv2.destroyAllWindows()
```

۲. دستور merge (ترکیب کانال‌های تصویر)

دستور merge کانال‌های جدا شده را دوباره به هم ترکیب می‌کند تا تصویر اصلی یا یک تصویر جدید تولید شود. مثلا، می‌توان کانال‌های جدا شده را دوباره ترکیب کرده یا کانال‌های مختلفی از تصاویر متفاوت را ترکیب نمود.

```
import cv2
image = cv2.imread('image.jpg')
                                                    جدا کردن کانال‌ها

b, g, r = cv2.split(image)
                                                    ترکیب دوباره کانال‌ها

merged_image = cv2.merge([b, g, r])
                                                    نمایش تصویر ترکیب شده

cv2.imshow('Merged Image', merged_image)
cv2.waitKey(۰)

cv2.destroyAllWindows()
```

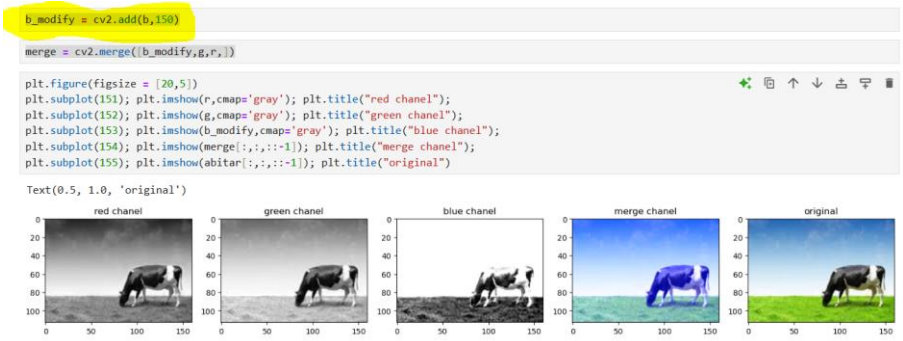
- در دستور split، کانال‌های رنگی تصویر به صورت مجزا برگردانده می‌شوند.
- در دستور merge، این کانال‌ها دوباره به تصویر رنگی تبدیل می‌شوند.

فصل دوم

پردازش تصویر و بینایی ماشین – بهبود کیفیت تصاویر

۲- فصل دوم: پردازش تصویر و بینایی ماشین – بهبود کیفیت تصاویر

۲- ۱- بهبود کانال‌های رنگی تصویر با استفاده از دستور add



شکل ۲- بهبود کانال‌های رنگی تصویر با استفاده از دستور add

کد بالا با استفاده از OpenCV و Matplotlib برای بارگذاری، تجزیه، و نمایش کانال‌های رنگی یک تصویر است. در ادامه به صورت مرحله به مرحله تفسیر می‌شود:

۱. بارگذاری تصویر

```
image = cv2.imread('x.jpg')
```

در اینجا، تصویر x.jpg با استفاده از OpenCV بارگذاری می‌شود. تابع cv2.imread() تصویر را به صورت یک آرایه عددی با ابعاد (عرض، ارتفاع، کانال‌ها) دریافت می‌کند که در آن هر کانال مربوط به یکی از رنگ‌های RGB (آبی، سبز، قرمز) است.

۲. جدا کردن کانال‌های تصویر

```
b, g, r = cv2.split(image)
```

در این مرحله، کانال‌های آبی (Blue)، سبز (Green) و قرمز (Red) تصویر با استفاده از تابع cv2.split() جدا می‌شوند. هر کانال یک ماتریس دو بعدی است که شدت مقادیر هر پیکسل در آن رنگ خاص را نشان می‌دهد.

۳. تغییر کانال آبی

```
b_modify = cv2.add(b, 150)
```

در این مرحله، به تمام مقادیر کانال آبی ۱۵۰ واحد اضافه می‌شود که باعث روشن‌تر شدن کانال آبی می‌گردد. از تابع cv2.add() برای جلوگیری از سرریز شدن اعداد استفاده می‌شود.

۴. ترکیب دوباره کانال‌ها

```
merge = cv2.merge([b_modify, g, r])
```

کانال‌های جدا شده (آبی اصلاح‌شده، سبز و قرمز) مجدداً با تابع cv2.merge() ترکیب می‌شوند تا یک تصویر رنگی جدید ایجاد شود.

۵. رسم کانال‌ها و تصاویر

```
plt.figure(figsize = [20,5])
plt.subplot(151); plt.imshow(r, cmap='gray'); plt.title("red channel");
plt.subplot(152); plt.imshow(g, cmap='gray'); plt.title("green channel");
plt.subplot(153); plt.imshow(b_modify, cmap='gray'); plt.title("blue channel");
plt.subplot(154); plt.imshow(merge[ : , : , : : -1]); plt.title("merge channel");
plt.subplot(155); plt.imshow(image[ : , : , : : -1]); plt.title("original")
```

این بخش از کد، تصاویر را به کمک Matplotlib در قالب چند زیرنمودار (subplot) نمایش می‌دهد :

- subplot(151) : کانال قرمز به صورت یک تصویر سیاه و سفید نمایش داده می‌شود. در نقشه رنگی cmap='gray'، شدت قرمز برای هر پیکسل نمایش داده می‌شود.
- subplot(152) : کانال سبز به همین صورت نمایش داده می‌شود.
- subplot(153) : کانال آبی که در مرحله قبل اصلاح شده است، نمایش داده می‌شود.
- subplot(154) : تصویر جدیدی که بعد از اصلاح کانال آبی و ترکیب کانال‌ها ایجاد شده است. توجه نمود که merge[-1 : : , ...] برای تبدیل از BGR به RGB به کار می‌رود زیرا OpenCV تصاویر را در قالب BGR ذخیره می‌کند اما Matplotlib آنها را به صورت RGB انتظار دارد.
- subplot(155) : تصویر اصلی بدون تغییر، برای مقایسه با تصویر ترکیب شده نمایش داده می‌شود.

این کد تصویری با کانال‌های جداگانه و یک تصویر ترکیب‌شده با کانال آبی روشن‌تر را نمایش می‌دهد. این تغییر در کانال آبی به وضوح باعث تغییر رنگ در تصویر ترکیب‌شده می‌شود که رنگ‌های حاوی آبی روشن‌تر دیده خواهند شد.

۲- ۲- چند سیستم رنگی رایج در پردازش تصویر و علوم کامپیوتر

در پردازش تصویر و علوم کامپیوتر، سیستم‌های رنگی مختلفی برای نمایش و پردازش رنگ‌ها استفاده می‌شوند. هر سیستم رنگی از ترکیب کانال‌های مختلف برای نمایش رنگ‌ها استفاده می‌کند. در اینجا چند سیستم رنگی رایج از جمله RGB، Grayscale، و HSV توضیح داده شده است:

۱. RGB (Red, Green, Blue)

سیستم رنگی RGB یکی از پرکاربردترین مدل‌های رنگی است که از سه کانال اصلی : قرمز (Red)، سبز (Green)، و آبی (Blue) تشکیل شده است. رنگ نهایی با ترکیب مقادیر این سه کانال به دست می‌آید.

- کاربرد : نمایشگرهای دیجیتال، عکاسی، پردازش تصویر
- مثال: (۰, ۰, ۲۵۵) نشان‌دهنده رنگ قرمز است، زیرا کانال قرمز در حداکثر مقدار خود (۲۵۵) است و کانال‌های سبز و آبی خاموش هستند (۰).

۲. Grayscale (تصویر خاکستری)

در این سیستم، تصویر به صورت تک کانالی نمایش داده می‌شود و هر پیکسل تنها یک مقدار شدت نور دارد که بین ۰ (سیاه) و ۲۵۵ (سفید) متغیر است. این سیستم فاقد رنگ است و تنها مقادیر روشنایی را نمایش می‌دهد.