آشنایی مقدماتی با

OpenCV

و برنامه نویسی با پایتون

عنوان : آشنایی مقدماتی با OpenCV و برنامه نویسی با پایتون
موضوع : بینایی ماشین
ترجمه و گردآوری : پیمان صالحی

اطلاعات بیشتر و دانلود نمونه کد ها در:
https://github.com/peymanslh/opencv-doc

پیشگفتار نویسنده

این کتابچه یک تحقیق دانشجویی در مورد کتابخانه قدرتمند OpenCV است و سعی بر این است تا خواننده در حد مقدماتی با این کتابخانه قدرتمند و پردازش تصویر آشنا شود و سر فصل های آن برای یادگیری کامل این کتابخانه مناسب نمی باشد. در بخش ابتدایی به طور مختصر دلیل استفاده از این کتابخانه در کنار تاریخچه آن شرح داده شده است و در بخش های بعد تنها چند تمرین کوتاه که با زبان پایتون نوشته شده است برای شروع مقدماتی با این کتابخانه آورده شده است. همچنین فایل تمرینی که به طور جداگانه نوشته شده است را میتوانید از صفحه گیت هاب این تحقیق دانلود کنید.

مقدمه

کتابخانه OpenCV یکی از مجبوب ترین کتابخانه هایی است که برای پردازش تصویر در نرم افزار ها به کار می رود. این کتابخانه این قدرت را به ما می دهد تا تعداد زیادی از الگوریتم های مختلف پردازش تصویر کامپیوتری را بصورت بلادرنگ اجرا کنیم. این کتابخانه طی سالهای طولانی گردآوری شده و یکی از کتابخانه های استاندارد در این زمینه است. یکی از مزیت های اصلی OpenCV بهینه سازی بالای آن است و در اکثر پلتفرم ها قابل دسترسی است.
در این کتابچه بیشتر به مقدمات و توضیحاتی در باب کاربرد های OpenCV و پردازش تصویر می پردازیم و در کنار نمونه کدهایی از این کتابخانه که به زبان پایتون نوشته می شوند نگاهی هم به یادگیری ماشین می اندازیم.

فهرست مطالب

بخش سوم

آشنایی بیشتر با OpenCV

منابع بیشتر برای یادگیری

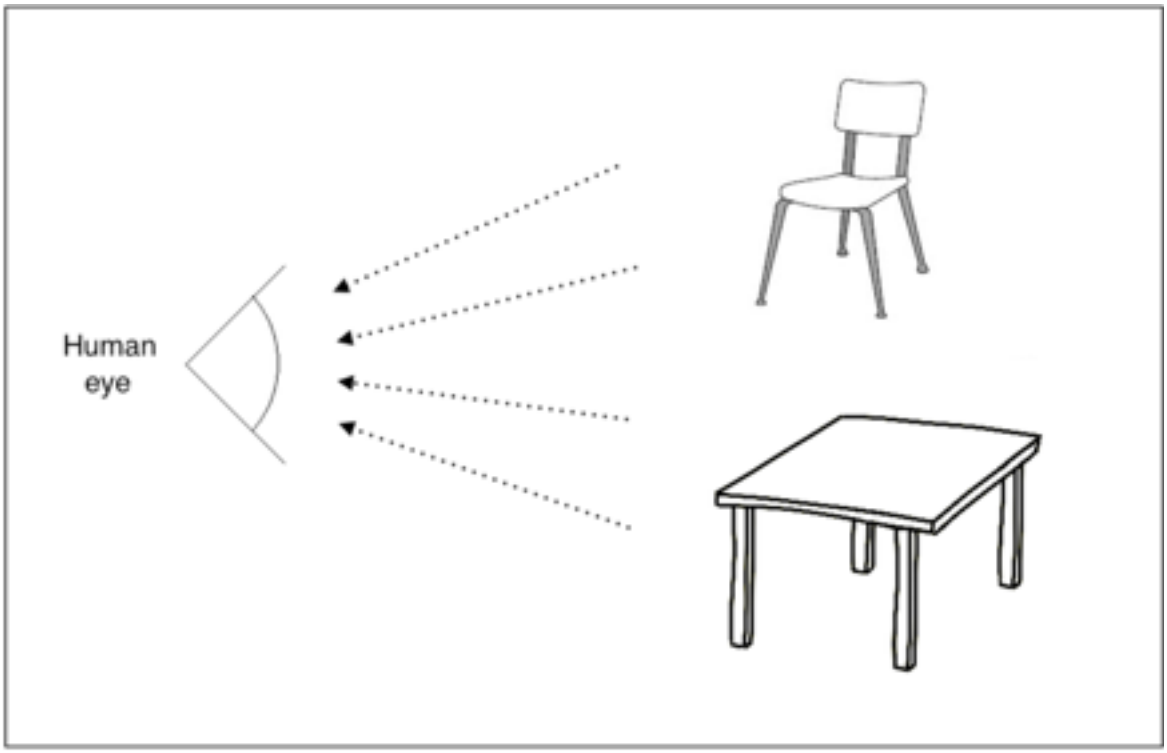
منابع

فصل اول:
آشنایی با OpenCV

نرم افزار های پردازش تصویر کامپیوتری جالب و کاربردی، اما الگوریتم های اصلی آن از لحاظ محاسباتی فشرده هستند. امروزه با ظهور پردازش ابری ما قدرت بیشتری برای کار با این نوع نرم افزار ها داریم. کتابخانه OpenCV به شما این امکان را میدهد تا الگوریتم های پردازش تصویر کامپیوتری را به طور مؤثر و بلادرنگ اجرا کنید. این کتابخانه سالهای زیادی است که ایجاد شده و در حال بروزرسانی است و در این زمینه یکی از کتابخانه های استاندارد به شمار می رود. یکی از مزیت های OpenCV بهینه سازی بالای آن است و در اکثر پلتفرم ها نیز قابل دسترسی است. در این کتابچه ما به مقدمات OpenCV می پردازیم و دلیل استفاده از این کتابخانه و روش استفاده از برخی الگوریتم های آن را شرح می دهیم.

سیستم بینایی انسان و پردازش تصویر در ماشین

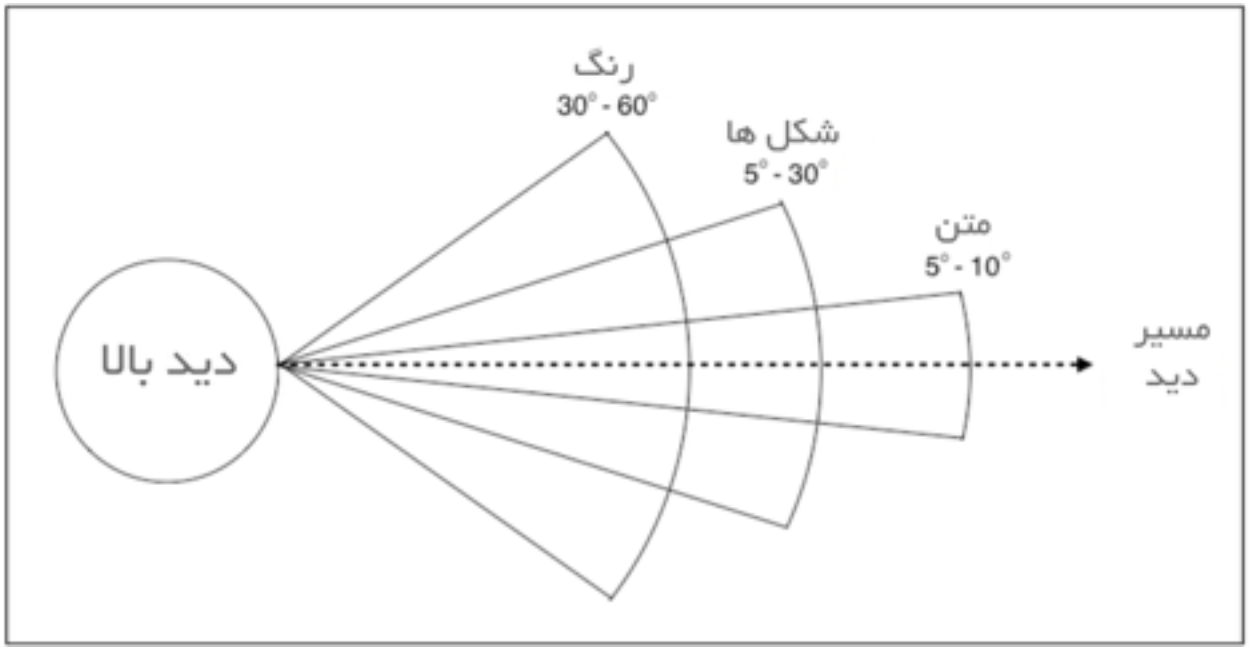
قبل از اینکه به سراغ OpenCV برویم اول باید بفهمیم چرا این توابع در OpenCV ایجاد شده است. یکی از نکات مهم قبل از توسعه درست الگوریتم های پردازش تصویر این است که شما بفهمید سیستم بینایی انسان چگونه کار می کند. هدف الگوریتم های پردازش تصویر فهمیدن محتوای تصاویر و ویدئو هاست. به نظر می رسد انجام این کار برای انسان ها به راحتی امکان پذیر است اما نکته مهم اینجاست که آیا ماشین ها هم می توانند این کار را با دقت و به راحتی انجام دهند؟
به تصویر زیر نگاه کنید:



چشم انسان تمامی اطلاعات رو به روی خود از جمله رنگ ها، شکل ها، روشنایی و غیره را می بیند. در تصویر قبل چشم انسان تمامی اطلاعات درباره دو شئ اصلی را به روش خاصی ذخیره می کند. اول از همه ما می فهمیم سیستم ما چگونه کار می کند، سپس با استفاده از این درک درست از سیستم بینایی خود میتوانیم به چیزی که می خواهیم برسیم. به عنوان مثال، چند چیز برای درک این موضوع نیاز است:

- سیستم بینایی ما به محتوای با فرکانس پایین نسبت به محتوای با فرکانس بالا حساس تر است، محتوای با فرکانس پایین به مناطق مسطحی که در آن مقادیر پیکسل ها به سرعت تغییر نمیکند و محتوای با فرکانس بالا به گوشه ها و لبه ها که در آن پیکسل ها به سرعت در حال نوسان است اشاره میکنند. شما به راحتی متوجه می شوید که آیا لکه ای روی سطحی مسطح وجود دارد یا نه اما به سختی میتوان فهمید که آیا روی سطحی شیار دار لکه ای وجود دارد یا خیر.
- چشم انسان به تغییرات در روشنایی نسبت به تغییرات در رنگ حساس تر است.
- سیستم بینایی ما در برابر حرکات حساس است. اگر چیزی در میدان دید ما حرکت کند ما به سرعت آن را تشخیص می دهیم حتی اگر مستقیما به آن خیره نباشیم.
- ما تمایل داریم تا یک یادداشت ذهنی از نکته ای برجسته از میدان دید خود بگیریم. بیایید یک میز سفید با چهار پایه سیاه و یک لکه قرمز در گوشه ای از سطح صاف ان را در نظر بگیریم. حالا به این میز نگاه کنید، شما بلافاصله یک یادداشت ذهنی از این میز که سطحی صاف، چهار پایه با رنگ مخالف و لکه ای قرمز که در گوشه ای از میز قرار گرفته در ذهن خود ایجاد می کنید. ذهن ما واقعا در این کار باهوش است! ما این عمل را به طور خودکار و بلافاصله انجام می دهیم و اگر دوباره با آن رو به رو شویم آن را به یاد می آوریم.

برای اینکه درکی از میدان دید انسان بگیریم، بیاید به دید بالای انسان و زاویه ای که ما میتوانیم چیز های مختلف را ببینیم بیندازیم:



سیستم بینایی ما قادر به دیدن چیز های بیشتری است، اما این گزینه ها برای شروع به نظر کافی می رسد. شما می توانید با جستجو در اینترنت موارد بیشتری را در رابطه با مدل های سیستم بینایی انسان بیابید.

چگونه انسان محتوای تصاویر را می فهمد؟

اگر شما نگاهی به اطراف خود بیندازید، اشیا زیادی می بینید. ممکن است هر روز با اشیا زیادی رو به رو شوید، و هر کدام از آن ها را بدون هیچ تلاشی تشخیص دهید. وقتی شما به یک صندلی نگاه می کنید، چند دقیقه برای تشخیص آن وقت صرف نمیکنید، در حقیقت این یک صندلی است و شما بلافاصله متوجه می شوید.

اساسا قسمت هایی در مغز ما وجود دارد که به ما در تشخیص اشیا کمک می کند. انسان ها می توانند اشیا مختلف را تشخیص دهند و اشیا شبیه به هم را دسته بندی کنند. ما با این دسته بندی ها و توسعه آن توسط اشیا مختلف به راحتی می توانیم این کار را انجام دهیم. وقتی ما به یک شئ نگاه می کنیم، ذهن ما نقطه های برجسته آن شئ را استخراج می کند که در این صورت عواملی مانند جهت، اندازه، چشم انداز و نور مهم نیستند.

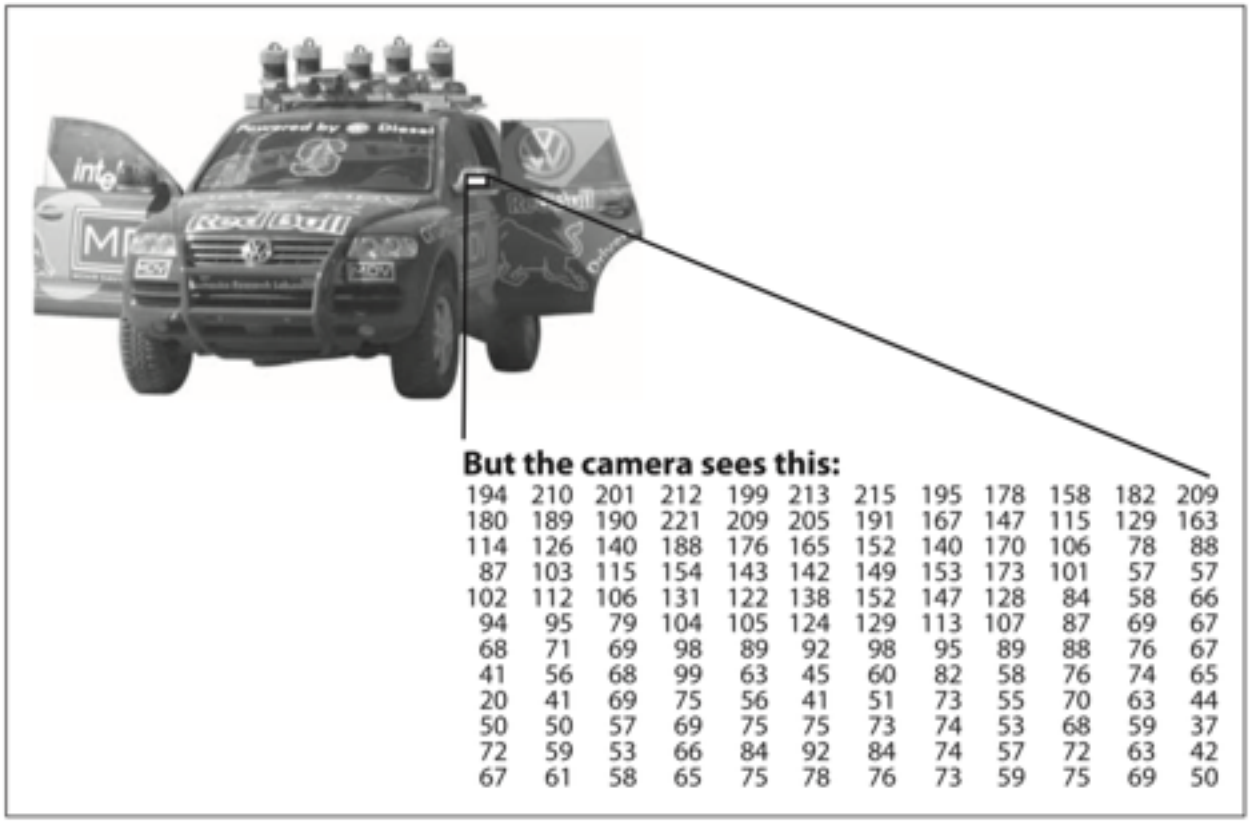
یک صندلی که دوبرابر سایز عادی خودش است و ۴۵ درجه چرخیده است همچنان یک صندلی است. ما به سبب روشی که برای تشخیص آن استفاده می کنیم به راحتی می توانیم ان را تشخیص دهیم. اما ماشین ها نمی توانند به راحتی این کار را انجام دهند. انسان ها اشیا را بر اساس شکل و مشخصات اصلی آن به خاطر می سپارند. صرف نظر از اینکه آن شئ چگونه قرار گرفته است، می توان آن را شناخت. در سیستم بینایی انسان، موقعیت، مقیاس و نقطه دید به قوی تر شدن ما کمک می کند.

اگر نگاه عمیق تری به سیستم خود بیندازیم، میبینید که انسان ها سلول هایی دارند که می تواند به اشکالی مانند خطوط منحنی و صاف عکس العمل نشان دهد. اگر بیشتر در امتداد سیستم بینایی خود حرکت کنیم، می توانیم مجموعه های بیشتری از سلول هایی را ببینیم که برای پاسخ دادن به اشیا بیشتری مانند درخت ها، در ها و غیره آموزش دیده اند. همچنین مغز ما برای افزایش تعداد این فیلد ها تمایل نشان می دهد.

حالا، از سوی دیگر، کامپیوتر ها این کار را به سختی انجام می دهند.

در یک سامانه ی بینایی ماشین، یک کامپیوتر، شبکه اي از اعداد را از دوربین یا دیسک سخت دریافت میکند. براي بسیاری قسمت ها، نه تشخیص الگوي درونی، نـه کنتـرل فوکـوس خودکـار پنجره و نه هیچ تناظر متقابل با سال ها تجربه وجود ندارد. براي بسیاری قسمت هـا، سـامانه هـای بینایی هنوز نسبتاً کودك هستند.

تصویر بعدی یک اتومبیل را نشان می دهد. در این تصویر، ما یک آینه بغـل را در سـمت راننده میبینیم. اما چیزي که کامپیوتر می بیند، تنها شبکه ای از اعداد است. هر عدد داده شده در این شبکه، مقدار نسبتاً بزرگی نویز دارد به طوري که به تنهایی اطلاعات کمی به ما میدهـد. امـا ایـن شبکه ی اعداد، همه ی آن چیزي است که کامپیوتر میبیند. در حقیقت وظیفه ما در قسمت پردازش تصویر در کامپیوتر ها تبدیل این اعداد به تصویر "آینه بغل" این اتومبیل است.



حالا فرض کنید که ماشین ما توانست این شبکه ی اعداد را به تصویری ۲ بعدی تبدیل کند و آن را نمایش دهد، اما حالا مشکل بزرگتری هم وجود دارد. ما باید اشیا سه بعدی که در دنیای واقعی وجود دارد و انسان نیز می تواند آن را به راحتی و با حس های مختلف از جمله بینایی، لامسه و غیره درک کند را به صورت تصویری دو بعدی که تنها از یک جهت گرفته شده و عواملی مانند تغییرات طبیعی (همچون آب و هوا، نور، انعکاس ها، جابه جایی ها) نواقص لنز و چیدمان مکانیکی، زمان کامل سازي محدود حسگر (ماتی ناشی از حرکت) نویز الکتریکی در حسگر یا دیگر قطعات الکترونیکی و مصنوعات ناشی از فشرده سازی پس از گرفتن عکس ناشی میشود را به ماشین بفهمانیم و از آن بخواهیم تا اشیائی که ما از آن انتظار داریم را از درون این تصویر دو بعدی تشخیص دهد و برایمان استخراج کند.

در طراحی یک سامانه ی عملی، اطلاعات زمینه ای دیگر می توانند براي کار با محدودیتهایی که حسگرهاي بصري به ما تحمیل میکنند، مورد استفاده قرار گیرند. به طور مثال یک ربات متحرك را درنظر بگیرید که بایستی در یک ساختمان، منگنه هایی را پیدا کرده و بردارد. ربات، ممکن است از این واقعیت استفاده کند که یک میز، شیء است که داخل ادارات پیدا میشود و منگنه ها معمولاً روي میز یافت میشوند که این یک مرجع اندازه ضمنی است. منگنه ها باید بتوانند روي میز جا بگیرند. این، همچنین کمک میکند تا منگنه هایی که در مکانهاي غیرمعمول مثلاً روي سقف یا پنجره تشخیص داده میشوند، حذف شوند. ربات میتواند به راحتی برخی اشیاء دیگر به شکل منگنه را که روي میز قرار نمیگیرند، نادیده بگیرد، زیرا این اشیا فاقد زمینه ی چوبی یک میز هستند. در کارهایی از قبیل بازیابی تصویر، تصاویر منگنه موجود در پایگاه داده ممکن است تصاویري باشند که در آنها اندازه منگنه ها کاملاً عادی بوده و به طور ضمنی فاقد برخی چیدمان های ممکن باشند. یعنی احتمالاً عکاس تنها از منگنه های واقعی با اندازه عادی عکس میگیرد. همچنین، اشخاص معمولاً تمایل دارند در هنگام

عکسبرداری، اشیا را در مرکز و در جهت های مشخصی قرار دهند. بنابراین، اغلب مقداری اطلاعات ضمنی غیرعمدی در تصاویرگرفته شده توسط افراد وجود دارد.
حال راهی که پیش روی ما قرار دارد این است که تمام ویژگی های یک شئ شامل اندازه، زاویه، چشم انداز و سایر اطلاعات را به صورت یک مجموعه آموزشی برچسب خورده در آوریم. اما این پروسه سخت و زمان بر است! همچنین ممکن نیست شما بتوانید اطلاعات هر چیزی را جمع آوری کنید. ماشین ها مقدار زیادی حافظه و مقداری زمان نیاز دارند تا این مدل ها را بسازند تا بتوانند اشیا را تشخیص دهند. حتی با همه اینها اگر شیئی به طور کامل مشخص نباشد کامپیوتر نمی تواند آن را تشخیص دهد و این به خاطر این است که کامپیوتر فکر می کند با یک شیئ جدید روبه رو شده است. پس ما به یک کتابخانه نیاز داریم تا همه ی این ویژگی ها را مد نظر بگیرد و با فرمول ها و الگوریتم های پردازش تصویر بتواند اشیا را تشخیص دهد. هدف ازOpenCV، فراهم آوردن ابزارهاي ابتدايی براي حل مسائل بینایی ماشین است. در بعضی موارد، توابع سطح بالا در کتابخانه براي حل مسائل بسیار پیچیده در بینایی ماشین کافی هستند. حتی زمانی که اینگونه نباشد، اجزاي پایه در کتابخانه به اندازه کافی براي ایجاد یک راه حل، کامل هستند.

تاریخچه OpenCV

OpenCV، از یک تحقیق ابتدایی اینتل براي توسعه کاربردهایی که از پردازنده به میزان زیادي استفاده میکنند، رشد کرد. براي رسیدن به این هدف، اینتل پروژه های بسیاري از قبیل ردگیري اشعه بیدرنگ و دیوارهاي نمایشگر سه بعدي را آغاز کرد. یکی از نویسندگان که در آن زمان براي اینتل کار میکرد، از چند دانشگاه دیدن میکرد و مشاهده کرد که چند گروه دانشگاهی برجسته از قبیل آزمایشگاه رسانه MIT به خوبی زیرساختهای بینایی ماشین متن باز را بین خود توسعه داده اند. کدي که از دانشجویی به دانشجوي دیگر انتقال داده میشد، نقطه شروع ارزشمندي براي آن دانشجو در توسعه برنامه بینایی خودش بود. به جاي دوباره نویسی توابع پایه یک دانشجوي جدید میتوانست از کدهاي قبل استفاده کند.
بنابراین OpenCV به عنوان راهی براي دراختیار گذاشتن عمومی زیربناي بینایی ماشین، متقاعد کننده به نظر میرسید. با کمک تیم کتابخانه اجرایی اینتل، OpenCV با یک هسته از کد پیاده سازي شده و خصوصیات الگوریتمی، به اعضاي تیم روسی اینتل فرستاده شد.
وادیم پیزاروسکی میان اعضاي تیم روسی، کسی است که مدیریت، کد کردن و بهینه سازی بسیاري از قسمتهاي OpenCV را انجام داده و هنوز در مرکز بسیاري از تلاشهاي OpenCV است. در کنار او ویکتور اروهیمو، به توسعه ساختار اولیه کمک کرد و والري کیوریاکین، مدیریت آزمایشگاه روسی را برعهده داشت و حمایت زیادي از این تلاش انجام داده است. OpenCV از ابتدا چند هدف را دنبال میکرد:

- توسعه ی تحقیقات بینایی از طریق فراهم کردن کد متن باز بهینه براي ساختار پایه بینایی به طوري که دیگر به بازآفرینی این چرخ احتیاجی نباشد.
- اشاعه دانش بینایی از طریق فراهم کردن یک زیرساخت مشترك به طوري که توسعه دهندگان بتوانند برنامه های خود را به کمک آن بسازند. بنابراین کدها خواناتر و قابلیت جابجایی بیشتري خواهند داشت.
- کاربردهاي تجاري مبتنی بر بینایی پیشرفته از طریق فراهم کردن کد بهینه، قابل حمل و رایگان، در حالی که نیاز نیست مجوز برنامه هاي نوشته شده خودشان متن باز یا رایگان باشند.

فراهم کردن کاربردهاي بینایی ماشین، نیاز به پردازندههاي سریع را افزایش میدهد و نیاز به ارتقا به پردازندههاي سریعتر، سود بیشتري نسبت به فروختن چند نرم افزار براي اینتل خواهد داشت. شاید

این دلیل آن باشد که چرا این کد متن باز و رایگان از یـک فروشـنده سـخت افـزاری برخاسته است و نه یک شرکت نرم افزاری. از اینرو میتوان دریافت که براي ابتکار در نرم افزار در یک شرکت سخت افزاری فضای بیشتری وجود دارد.

کاربردهای OpenCV

با استفاده از OpenCV میتوانید عملیات های زیر را به راحتی انجام دهید:

- عملیات های پردازش تصویر
- ساخت GUI
- آنالیز ویدئو ها
- بازسازی سه بعدی
- استخراج ویژگی ها
- تشخیص شی
- یادگیری ماشین
- عکاسی محاسباتی
- آنالیز اشکال
- الگوریتم های جریان بصری
- شناخت چهره و اشیا
- تطبیق سطوح
- تشخیص و شناخت متن

شروع کار با OpenCV

نصب OpenCV

از آنجایی که نصب OpenCV برای سیستم عامل های مختلف روش های متفاوتی دارد و ممکن است تغییر کند پیشنهاد میکنیم برای نصب به صفحه دانلود OpenCV در وبسایت رسمی آن به آدرس زیر مراجعه کنید.

http://opencv.org/downloads.html

ویژگی های Gui در OpenCV

خواندن تصویر

برای خواندن تصاویر با استفاده از OpenCV می توانید از تابع ()cv2.imread استفاده کنید.
مقادیری که میتوانید برای این تابع بفرستید به شرح زیر است.
cv2.IMREAD_COLOR: فراخوانی تصاویر به صورت رنگی
cv2.IMREAD_GRAYSCALE : فراخوانی تصاویر بصورت سیاه و سفید
cv2.IMREAD_UNCHANGED : فراخوانی تصاویر بدون تغییر
همچنین بجای استفاده از این توابع می توانید به ترتیب اعداد ۰ ، ۱ و -۱ را به تابع ()imread دهید.
مانند مثال زیر:

```
import numpy as np
import cv2
# Load an color image in grayscale
img = cv2.imread('messi5.jpg',0)
```

نمایش تصویر

برای نمایش تصویر می توان از تابع ()imshow استفاده کرد.
آرگومان اول نام پنجره ای که باز می شود است و باید بصورت رشته بنویسید و آرگومان دوم تصویر مورد نظر را می گیرد. شما میتوانید چند پنجره ایجاد کنید اما باید از نام های متفاوت استفاده کنید.

```
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

تابع ()waitKey یک عدد را در واحد میلی ثانیه دریافت میکند و اگر طی آن مدت شما کلیدی از کیبرد را نزنید دستور بعدی را اجرا می کند. فرستادن 0 به معنی بی نهایت می باشد.
تابع ()destroyAllWindows تمامی پنجره ها را می بندد. همچنین میتوانید با انتخاب پنجره ای خاص فقط همان پنجره را ببندید.

```
cv2.namedWindow('image', cv2.WINDOW_NORMAL)
cv2.imshow('image',img)
```

```
cv2.waitKey(0)
cv2.destroyAllWindows()
```

ذخیره تصویر

برای ذخیره تصویر می توانید از تابع imwrite() استفاده کنید. کد زیر تصویر مورد نظر را با پسوند png. ذخیره می کند.

```
cv2.imwrite('messigray.png',img)
```

جمع بندی

در مثال زیر در صورتی که کلید ESC را بفشارید برنامه را می بندد و در صورتی که کلید ''S را بفشارید تصویر را به صورت سیاه سفید و با پسوند png. ذخیره میکند و سپس پنجره ها را میبندد.

```
import numpy as np
import cv2
img = cv2.imread('messi5.jpg',0)
cv2.imshow('image',img)
k = cv2.waitKey(0)
if k == 27:         # wait for ESC key to exit
    cv2.destroyAllWindows()
elif k == ord('s'): # wait for 's' key to save and exit
   cv2.imwrite('messigray.png',img)
   cv2.destroyAllWindows()
```

رسم شکل

در فانکشن های زیر میتوانید آرگومان های زیر را ارسال کنید:

img : تصویری که میخواهید اشکال را روی آن رسم کنید.

color : رنگ شکلی که میخواهید رسم کنید، به صورت BGR باید بفرستید مانند (255,0,0) برای رنگ آبی.

thickness : برای تنظیم ضخامت شکل، اگر مقدار -۱ را ارسال کنید و شکل شما دایره یا مستطیل باشد به صورت تو پر رنگ می شود همچنین مقدار پیش فرض عدد ۱ است.

lineType : نوع خط.

رسم خط

برای رسم خط باید مختصات نقطه ابتدا و انتها را به تابع cv2.line() بفرستیم. در مثال زیر ابتدا یک تصویر سیاه ایجاد و خطی آبی رنگ روی ان رسم میکنیم.

```
import numpy as np
import cv2

# Create a black image
img = np.zeros((512,512,3), np.uint8)

# Draw a diagonal blue line with thickness of 5 px
cv2.line(img,(0,0),(511,511),(255,0,0),5)
```

رسم چهارگوشه

برای رسم مربع یا مستطیل باید مختصات گوشه بالا-چپ و پایین-راست را بفرستیم. در این مثال مربعی در گوشه بالا-راست تصویر با رنگ سبز میکشیم.

```
cv2.rectangle(img,(384,0),(510,128),(0,255,0),3)
```

رسم دایره

برای رسم دایره باید مختصات مرکز دایره و شعاع آن را بفرستید. در مثال زیر دایره ای در گوشه بالا-راست تصویر با رنگ قرمز و توپر رسم میکنیم.

```
cv2.circle(img,(447,63), 63, (0,0,255), -1)
```

رسم بیضی

برای رسم بیضی باید چند مقدار را بفرستیم، مقدار اول برای مرکز بیضی، مقدار دوم برای طول محور های اصلی و دوم، زاویه چرخش بیضی بر خلاف عقربه ساعت، startAngle و endAngle نشان دهنده قوس بیضی در جهت عقربه ساعت. مثال زیر یک بیضی به صورت نیمه در وسط تصویر رسم میکند.

```
cv2.ellipse(img,(256,256),(100,50),0,0,180,255,-1)
```

رسم چند ضلعی

برای رسم چند ضلعی باید مختصات راس ها را داشته باشید، این نقاط را باید به صورت آرایه ای از اشکال به صورت ROWSx1x2 بفرستیم. در مثال زیر یک چند ضلعی با چهار رأس و با رنگ زرد رسم میکنیم.

```
pts = np.array([[10,5],[20,30],[70,20],[50,10]], np.int32)
pts = pts.reshape((-1,1,2))
```

```
cv2.polylines(img,[pts],True,(0,255,255))
```

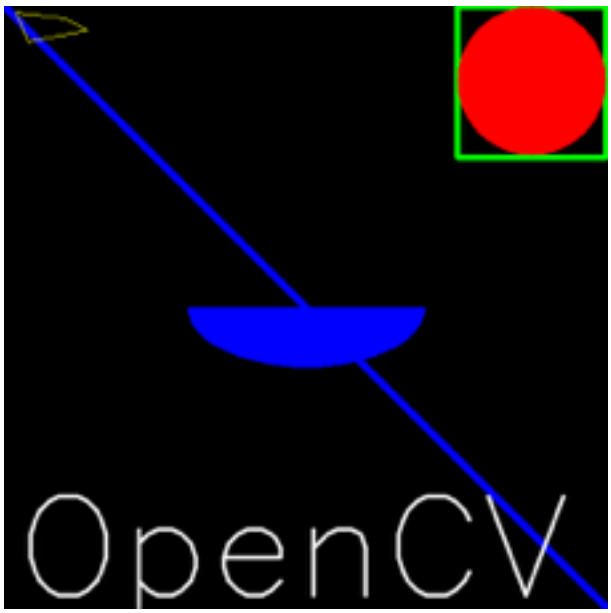
اضافه کردن متن به تصویر

برای اضافه کردن متن به تصویر به مقادیر زیر نیاز دارید.

- متنی که میخواهید بنویسید.
- موقعیت متن در تصویر
- نوع فونت
- مقیاس فونت (اندازه فونت را مشخص میکند)
- و مقادیری که در مثال های قبل توضیح دادیم، مانند ضخامت، رنگ و ...

```
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,'OpenCV',(10,500), font, 4,(255,255,255),
2,cv2.LINE_AA)
```

در نهایت تصویر پایانی ما به شکل زیر می باشد.



عملیات های ابتدایی بر روی تصویر

دسترسی به پیکسل ها

شما میتوانید رنگ یک پیکسل را با دادن مختصات افقی و عمودی آن پیکسل به دست آورید. این مقدار به صورت BGR به شما برگردانده می شود.

```
import cv2
import numpy as np
img = cv2.imread('messi5.jpg')
px = img[100,100]
print px
// [157 166 200]
```

```
# accessing only blue pixel
blue = img[100,100,0]
print blue
// 157
```
همچنین میتوانید رنگ یک پیکسل را با دادن مقدار جدید به آن تغییر دهید.
```
img[100,100] = [255,255,255]
print img[100,100]
// [255 255 255]
```
دسترسی به مشخصات تصویر
برای دسترسی به مشخصات تصویر می توانید از توابع زیر استفاده کنید.
```
print img.shape
// (300, 500, 3)
```
این تابع آرایه ای را بر میگرداند که مقدار اول آن تعداد ردیف های تصویر، عدد دوم تعداد ستون های تصویر و عدد سوم اگر تصویر شما رنگی باشد و از نوع grayscale نباشد تعداد کانال های رنگی تصویر را بر میگرداند که برای تصاویر رنگی ۳ می باشد.
```
print img.size
// 450000
```
این تابع تعداد کل پیکسل ها را بر میگرداند. در تصویر ما که تعداد ستون ها ۵۰۰ و تعداد ردیف ها ۳۰۰ و تعداد کانال ها ۳ می باشد عدد به دست آمده ۴۵۰۰۰ می شود.
```
print img.dtype
// uint8
```
این تابع نیز نوع داده را بر میگرداند.

ایجاد خط دور تصویر (border)
برای ایجاد خط دور تصویرمان میتوانیم از توابعی که در مثال زیر آمده استفاده کنیم. مقادیری که باید برای تابع زیر بفرستیم به شکل زیر است.

src : تصویری که قرار است تغییرات بر روی آن انجام شود.

top, bottom, left, right : ضخامت خط دور را در جهت های بالا، پایین، چپ و راست را میتوان با این گزینه بر اساس پیکسل مشخص کرد.

cv2.BORDER_CONSTANT : یک خط تک رنگ به دور تصویر میکشد.

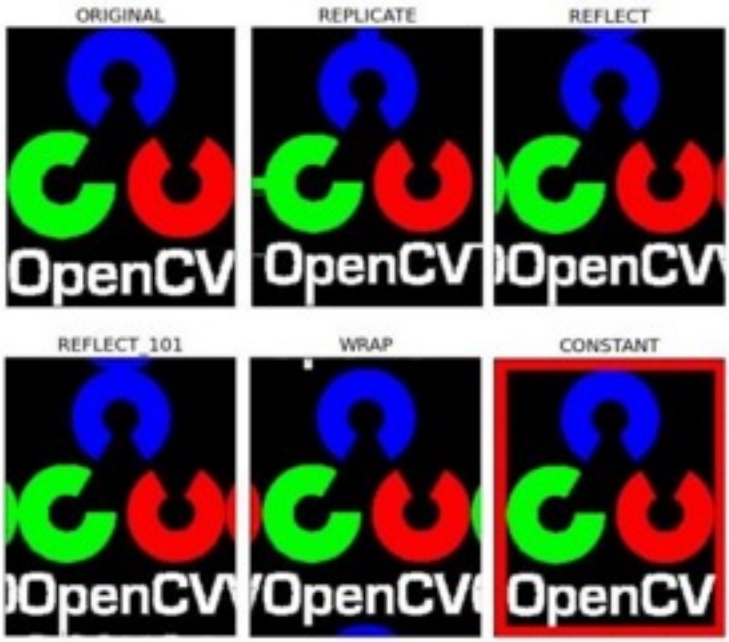
cv2.BORDER_REFLECT : همان قسمتی که خط قرار است ایجاد شود را برعکس میکند و مثل آینه عمل میکند، مانند: fedcba|abcdefgh|hgfedcb

cv2.BORDER_REFLECT_101 : مانند قبلی عمل میکند ولی با کمی تغییر، مانند: gfedcb|abcdefgh|gfedcba

cv2.BORDER_REPLICATE : اولین و آخرین مقدار تکرار میشود، مانند : aaaaaa|abcdefgh|hhhhhhh

cv2.BORDER_WRAP : نمیشود توضیح داد، فقط مثال را ببینید: cdefgh|abcdefgh|abcdefg

value : اگر خط از نوع cv2.BORDER_CONSTANT باشد میتوان با این گزینه رنگی را برای آن فرستاد.




```
import cv2
import numpy as np

BLUE = [255,0,0]
img1 = cv2.imread('opencv_logo.png')

replicate =
cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER_REPLICATE)
reflect =
cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER_REFLECT)
reflect101 =
cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER_REFLECT_101)
wrap = cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER_WRAP)
constant=
cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER_CONSTANT,value=BL
UE)
```

ترکیب تصاویر

شما با استفاده از OpenCV میتوانید دو تصویر را با استفاده از تابع cv2.addWeighted() با هم ترکیب کنید و آن ها را روی هم قرار دهید و برای هر کدام سطح شفافیت مشخص کنید.

```
img1 = cv2.imread('ml.png')
img2 = cv2.imread('opencv_logo.jpg')

dst = cv2.addWeighted(img1,0.7,img2,0.3,0)
```

عملیات های بیتی

عملیات های بیتی شامل AND, OR, NOT و XOR می شوند و در مواقعی که میخواهید قسمتی از تصویر را استخراج کنید به کار می آیند. در مثال بعدی ما می خواهیم قسمتی از یک تصویر را جدا کنیم و بر روی تصویر بعدی قرار دهیم. به مثال زیر توجه کنید.

```
# Load two images
img1 = cv2.imread('messi5.jpg')
img2 = cv2.imread('opencv_logo.png')

# I want to put logo on top-left corner, So I create a ROI
rows,cols,channels = img2.shape
roi = img1[0:rows, 0:cols ]

# Now create a mask of logo and create its inverse mask also
img2gray = cv2.cvtColor(img2,cv2.COLOR_BGR2GRAY)
ret, mask = cv2.threshold(img2gray, 10, 255, cv2.THRESH_BINARY)
mask_inv = cv2.bitwise_not(mask)

# Now black-out the area of logo in ROI
img1_bg = cv2.bitwise_and(roi,roi,mask = mask_inv)

# Take only region of logo from logo image.
img2_fg = cv2.bitwise_and(img2,img2,mask = mask)

# Put logo in ROI and modify the main image
dst = cv2.add(img1_bg,img2_fg)
img1[0:rows, 0:cols ] = dst

cv2.imshow('res',img1)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

به تصویر زیر دقت کنید. تصویر سمت چپ تصویر لایه ماسکی که ایجاد کردیم را نشان می دهد و تصویر سمت راست خروجی نهایی.

پردازش تصویر در OpenCV

تطبیق الگو ها

با استفاده از تابع cv2.matchTemplate() می توان تصویری را به عنوان الگو انتخاب کرد و در تصویری بزرگتر آن را جستجو کرد، هر گاه نمونه ای پیدا شد مختصات آن را بر میگرداند.
در مثال زیر نمونه ای از تصویر مسی را به عنوان الگو قرار می دهیم و در تصویری دیگر آن را جستجو میکنیم.

```
import cv2
import numpy as np
img = cv2.imread('messi5.jpg',0)
img2 = img.copy()
template = cv2.imread('template.jpg',0)
w, h = template.shape[::-1]
# All the 6 methods for comparison in a list
methods = ['cv2.TM_CCOEFF', 'cv2.TM_CCOEFF_NORMED',
'cv2.TM_CCORR','cv2.TM_CCORR_NORMED', 'cv2.TM_SQDIFF',
'cv2.TM_SQDIFF_NORMED']
for meth in methods:
   img = img2.copy()
   method = eval(meth)
   # Apply template Matching
   res = cv2.matchTemplate(img,template,method)
   min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
   # If the method is TM_SQDIFF or TM_SQDIFF_NORMED, take
minimum
```

```
if method in [cv2.TM_SQDIFF, cv2.TM_SQDIFF_NORMED]:
    top_left = min_loc
else:
    top_left = max_loc
bottom_right = (top_left[0] + w, top_left[1] + h)
cv2.rectangle(img,top_left, bottom_right, 255, 2)
cv2.imshow('result', res)
cv2.imshow('img', img)
```

نتیجه کدهای بالا را با متد های مختلف را میتوانید ببینید.

- cv2.TM_CCOEFF

- cv2.TM_CCOEFF_NORMED

- cv2.TM_CCORR

- cv2.TM_CCORR_NORMED

- cv2.TM_SQDIFF

- cv2.TM_SQDIFF_NORMED

جدا کردن شیئ از تصویر با الگوریتم GrabCut

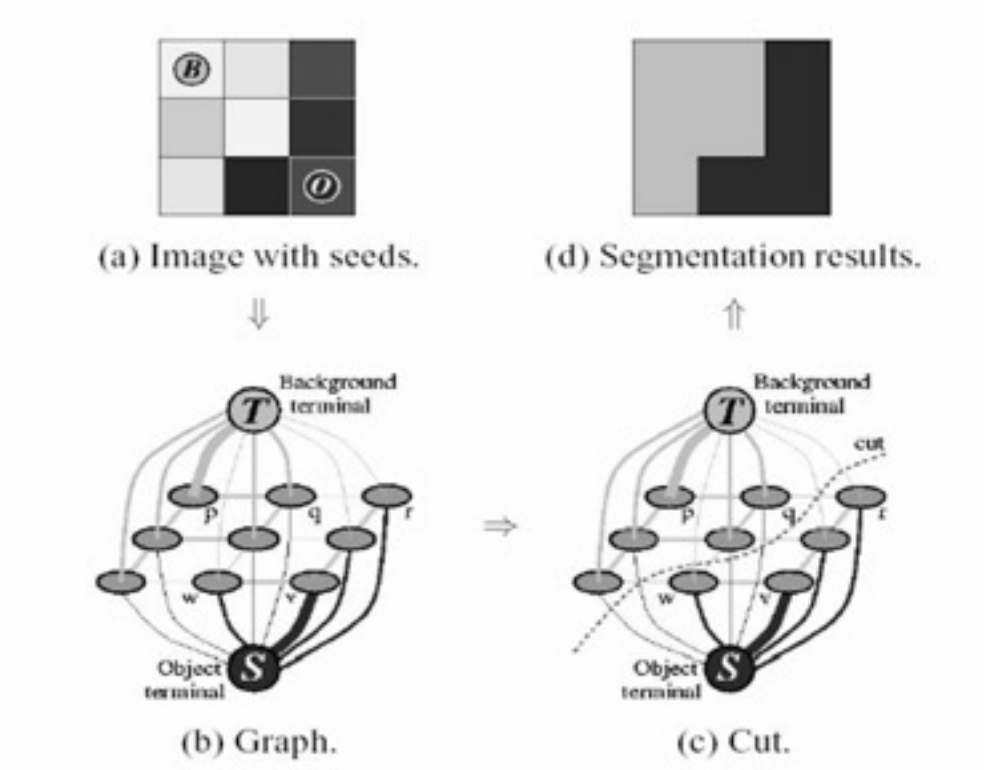
طرز کار این الگوریتم به این صورت است که شما فضایی را با یک چهار گوشه مشخص می کنید و الگوریتم با تکرار سعی میکند شیئ داخل این چهار گوشه را به بهترین شکل جدا کند. البته در بعضی مواقع ممکن است نتیجه زیاد خوب نباشد.

به تصویر بالا دقت کنید. ابتدا چهار گوشه با خط آبی مشخص شده و سپس قسمت هایی از بدن بازیکن و توپ که باید در تصویر بماند با رنگ سفید مشخص شده اند و قسمت هایی مانند آرم تبلیغاتی در گوشه سمت راست پایین که باید حذف شوند با رنگ مشکی مشخص شده اند. خروجی را میتوانید در سمت راست ببینید.

اما در پس زمینه چه اتفاقی می افتد؟

کاربر ورودی را به عنوان چهار گوشه مشخص میکند و هر چیزی خارج از این چهار گوشه باشد به عنوان پس زمینه شناخته می شود. حالا همه چیز در داخل مستطیل ناشناخته است. تمامی قسمت هایی که کاربر مشخص کرده از جمله پس زمینه و پیش نما به عنوان ناشناخته علامت گذاری میشوند و این به این معنی است که در فرآیند تغییر نمیکنند.

کامپیوتر داده هایی که ما دادیم (پیکسل های پس زمینه و پیش نما) را برچسب گذاری میکند.
حالا از Gaussian Mixture Model برای مدل سازی پس زمینه و پیش نما استفاده می شود.
حالا GMM با توجه به پیکسل های پس زمینه سعی میکند پیکسل های شبیه به آن در پیش نما را تشخیص دهد.
یک گراف از توزیع پیکسل ها ساخته می شود. گره ها در گراف پیکسل ها هستند. دو گره اضافه میشوند، گره منبع و گره نزول. هر پیکسل از پیش نما به گره منبع متصل میشود و هر گره از پیکسل پس زمینه به گره نزول.
تصویر زیر نمایانگر طرز کار این الگوریتم است.




```
import numpy as np
import cv2
img = cv2.imread('messi5.jpg')
mask = np.zeros(img.shape[:2],np.uint8)
bgdModel = np.zeros((1,65),np.float64)
fgdModel = np.zeros((1,65),np.float64)
rect = (50,50,450,290)
cv2.grabCut(img,mask,rect,bgdModel,fgdModel,
5,cv2.GC_INIT_WITH_RECT)
```

```
mask2 = np.where((mask==2)|(mask==0),0,1).astype('uint8')
img = img*mask2[:,:,np.newaxis]
cv2.imshow('result', img)
```

مثال بالا خروجی زیر را تولید می کند.

در تصویر بالا موهای سر بازیکن برداشته شد و همچنین آرم تبلیغاتی و چمن های زیر پای بازیکن نیز باید حذف شوند.

برای این کار ما تصویری جدید با همین اندازه با رنگ مشکی ایجاد میکنیم و قسمت های را که نباید حذف شوند با رنگ سفید رنگ میکنیم و قسمت هایی که باید حذف شوند را با رنگ خاکستری مشخص میکنیم. سپس از این تصویر به عنوان ماسک برای بهتر کردن خروجی تصویرمان استفاده میکنیم.

```
# newmask is the mask image I manually labelled
newmask = cv2.imread('newmask.png',0)

# whereever it is marked white (sure foreground), change mask=1
# whereever it is marked black (sure background), change mask=0
mask[newmask == 0] = 0
mask[newmask == 255] = 1
mask, bgdModel, fgdModel =
cv2.grabCut(img,mask,None,bgdModel,fgdModel,
5,cv2.GC_INIT_WITH_MASK)
mask = np.where((mask==2)|(mask==0),0,1).astype('uint8')
img = img*mask[:,:,np.newaxis]
cv2.imshow('result', img)
```

خروجی نهایی شبیه تصویر زیر می شود.

تشخیص چهره و اشیا

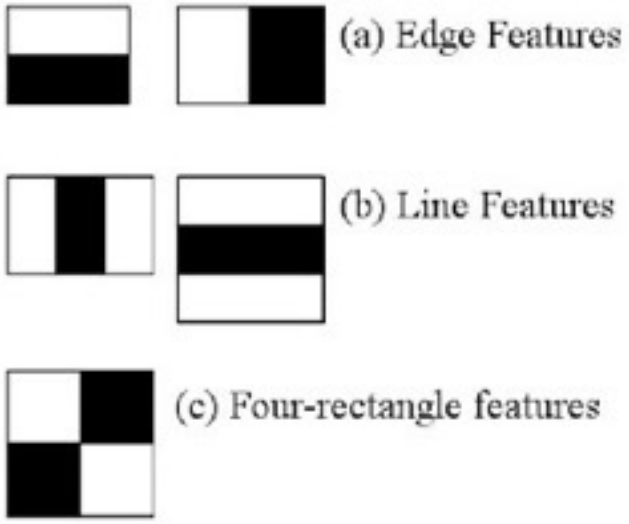
یادگیری ماشین (Machine Learning) در OpenCV

هدف یادگیری ماشین تبدیل داده ها به اطلاعات است. بعد از اینکه یک سری از داده ها را به کامپیوتر دادیم از آن انتظار داریم تا به سوال هایی که پیرامون آن موضوع میپرسیم پاسخ دهد. با توجه به داده هایی که ما به کامپیوتر دادیم از آن انتظار داریم تا به عنوان مثال به ما بگوید آیا ماشینی در تصویر می بیند یا خیر؟ یادگیری ماشین تبدیل داده ها به اطلاعات با استخراج قواعد و الگو ها از داده هاست.
با استفاده از یادگیری ماشین میتوان کارهایی مثل مقادیر دما، پیش قیمت سهام و غیره را پیش بینی کرد.
به عنوان مثال میتوان اطلاعات ۱۰،۰۰۰ تصویر را گرفت و تمام جزئیات آنها را تجزیه و تحلیل کرد از جمله اندازه گیری مقادیری مانند جهت لبه های صورت، پهنا و ارتفاع و فاصله مرکز صورت تا لبه ها و غیره. ما میتوانیم حدود ۵۰۰ نکته از جزئیات هر تصویر را بگیریم و این اطلاعات را با الگوریتم های یادگیری ماشین تجزیه و تحلیل کنیم. همچنین میتوانیم با دادن مقدار سن هر چهره و مقایسه آن ها با هم سن هر فرد را پیش بینی کنیم.
الگوریتم های یادگیری ماشین پشتیبانی شده در OpenCV :

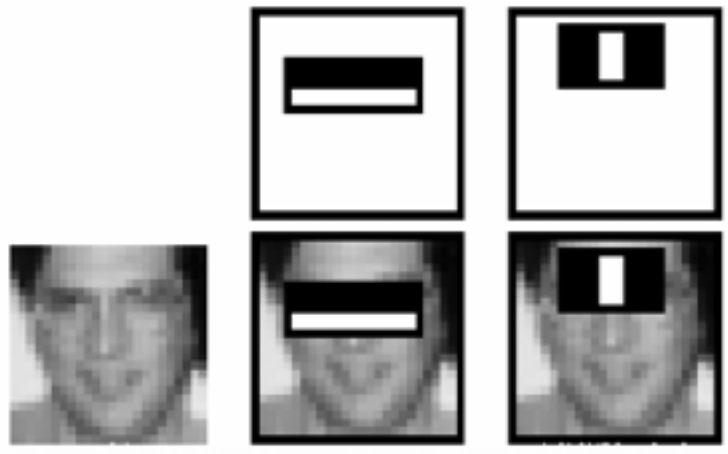
- Mahalanobis
- K-means
- Normal/Naïve Bayes classi er
- Decision trees
- Boosting
- Random trees
- Face detector / Haar classifier
- Expectation maximization (EM)
- K-nearest neighbors
- Neural networks / Multilayer perceptron (MLP)
- Support vector machine (SVM)

الگوریتم Face detector / Haar classifier

تشخیص اشیا با استفاده از Haar cascade یک روش موثر تشخیص شئ است که Paul Viola و Michael Jones در مقاله خود پیشنهاد دادند. روش "تشخیص سریع شیء با استفاده از یک آبشار افزایشی از ویژگی های ساده" در سال ۲۰۰۱ شروع شد. این روش یک رویکرد مبتنی بر یادگیری ماشین است که با استفاده از تعداد زیادی از تصاویر مثبت و منفی آموزش دیده است و سپس برای تشخیص اشیا در تصاویر دیگر به کار گرفته شد.
در این قسمت ما با تشخیص چهره کار خواهیم کرد. در ابتدا الگوریتم نیاز به تعداد زیادی تصاویر مثبت (تصاویر چهره) و تصاویر منفی (تصاویر بدون چهره) دارد تا بتوانیم اطلاعات را طبقه بندی کنیم.
سپس باید جزئیات را از تصاویر استخراج کنیم. برای این کار از ویژگی Haar نشان داده شده در تصاویر زیر استفاده میکنیم. هر ویژگی یک ارزش واحد است که با کم کردن مجموع پیکسل های زیر مستطیل سفید از مجموع پیکسل های زیر مستطیل سیاه به دست می آید.



حالا هر سایز و موقعیت ممکن از هر هسته برای محاسبه تعداد زیادی از ویژگی ها به کار میرود. (فقط تصور کنید چند محاسبه برای این کار لازم است؟ برای یک پنجره ۲۴*۲۴ چیزی حدود ۱۶۰۰۰۰ ویژگی!) برای هر ویژگی ما به محاسبه مجموع تعداد پیکسل های زیر مستطیل های سفید و سیاه نیاز داریم. برای حل این مشکل تصاویر انتگرالی معرفی شد و به راحتی مجموع پیکسل ها را محاسبه میکند.
اما از بین اینهمه ویژگی که محاسبه شد تنها چند ویژگی مفید میباشد. برای مثال در تصویر زیر در ردیف اول دو ویژگی نشان داده شده است. در ویژگی اول به نظر میاید تمرکز ویژگی مشخص شده بیشتر به ناحیه چشم است که بیشتر مواقع تیره تر از ناحیه بینی و گونه است و ویژگی دوم تمرکزش بیشتر بر روی چشم هاست که تیره تر شده است تا روی بینی که بین دو چشم قرار گرفته است. اما ویژگی هایی که تمرکزش بر روی گونه یا قسمت های دیگر است به نظر میرسد بی ربط است. پس ما چگونه باید از بین بیش از ۱۶۰۰۰۰ ویژگی بهترین ها را انتخاب کنیم؟ جواب این مشکل الگوریتم Adaboost است.



آدابوست مخفف بوستینگ تطبیقی بوده و یک الگوریتم یادگیری ماشین است که توسط یاو فروند و رابرت شاپیر ابداع شد. در واقع آدابوست یک متا الگوریتم است که بمظور ارتقاء عملکرد، و رفع مشکل رده‌های نامتوزان همراه دیگر الگوریتم‌های یادگیری استفاده می‌شود. در این الگوریتم، طبقه بند هر مرحله جدید به نفع نمونه‌های غلط طبقه‌بندی شده در مراحل قبل تنظیم می‌گردد. آدابوست نسبت به داده‌های نویزی و پرت حساس است؛ ولی نسبت به مشکل بیش برازش از بیشتر الگوریتم‌های یادگیری برتری دارد. طبقه بند پایه که در اینجا استفاده می‌شود فقط کافیست از طبقه بند نصادفی(۵۰٪) بهتر باشد و به این ترتیب بهبود عملکرد الگوریتم با تکرارهای بیشتر بهبود می‌یابد. حتی طبقه بندهای با خطای بالاتر از تصادفی با گرفتن ضریب منفی عملکرد کلی را بهبود می‌بخشند. در الگوریتم آدابوست در هر دور $t = 1,\ldots,T$ یک طبقه بند ضعیف اضافه می‌شود. در هر فراخوانی بر اساس اهمیت نمونه‌ها، وزن‌ها D_1 بروز می‌شود. در هر دور وزن نمونه‌های غلط طبقه‌بندی شده افزایش و وزن نمونه‌های درست طبقه‌بندی شده کاهش داده می‌شود؛ بنابراین طبقه بند جدید تمرکز بر نمونه‌هایی که سخت تر یادگرفته می‌شوند، خواهند داشت.

حالا ما ویژگی ها را بر روی تصاویر تمرینی اعمال می کنیم و او تصویر های مثبت و منفی را برای ما پیدا میکند.

فرآیند به این شکل است که همه تصاویر با وزن یکسان وارد می شوند و بعد از هر طبقه بندی وزن تصاویر نادرست افزایش یافته و دوباره این روند تکرار می شود. نرخ خطای جدید محاسبه میشود. وزن های جدید محاسبه میشود. این کار تا نرخ خطا پایین بیاید و یا ویژگی های مورد نیاز یافت شود ادامه میابد.

در طبقه بندی نهایی مجموع وزن این طبقه بندی ظعیف است. ضعیف است زیرا به تنهایی کافی نیست ولی همراه با دیگر طبقه بندی ها قدرتمند میشود. این مقاله میگوید ۲۰۰ ویژگی با دقت ۹۵٪ به دست آمده است. در جمع بندی نهایی چیزی حدود ۶۰۰۰ ویژگی میشود و این یعنی از بیش از ۱۶۰۰۰۰ به ۶۰۰۰ رسیدیم.

حالا تصویری را دریافت میکنیم و نگاهی به هر ۲۴*۲۴ پنجره می اندازیم. تمام ۶۰۰۰ ویژگی را روی آن اجرا میکنیم و چک میکنیم که آیا چهره ای را تشخیص داد یا نه؟ به نتیجه ی خیره کننده ای میرسیم و در زمان کمی توانستیم این کار را انجام دهیم.

در یک تصویر در تمام قسمت ها چهره وجود ندارد و این ایده ی خوبی است که ما تابعی داشته باشیم تا اول چک کند که آیا یک پنجره حاوی تصویر چهره است یا خیر، و اگر نبود از آن قسمت بگذریم تا در دفعات بعد دوباره آن را پردازش نکنیم و تمرکز را بر روی قسمت هایی که حاوی چهره است قرار دهیم. با این روش ما به صورت چشم گیری در زمان صرفه جویی میکنیم.

در مرحله بعد برای اینکه تمام ۶۰۰۰ ویژگی را بر روی یک پنجره اعمال نکنیم ویژگی ها را به طبقه بندی های کوچکتر تقسیم میکنیم. اول مرحله ای با ویژگی های کمتر، اگر از این مرحله عبور کرد به مرحله بعد میرود و اگر نه که هیچ. در مرحله بعد با ویژگی های بیشتر روبه رو میشود و اگر از این مرحله عبور کرد به مراحل بعد با ویژگی های بیشتر میرود.

نویسندگان این الگوریتم میگویند ۶۰۰۰ ویژگی در ۳۸ مرحله قرار میگیرد که به ترتیب در ۵ مرحله اول با ۱، ۱۰، ۲۵، ۲۵ و ۵۰ ویژگی است.(۲ ویژگی که در تصویر قبل نشان داده شده به عنوان کاربردی ترین ویژگی ها توسط Adaboost معرفی شده است.) به گفته نویسندگان به طور متوسط ۱۰ ویژگی از ۶۰۰۰ ویژگی در هر زیر پنجره ارزیابی شده است.

این یک توضیح مختصر درباره الگوریتم تشخیص چهره Viola-Jones بود. برای کسب اطلاعات بیشتر میتوانید به فصل منابع بیشتر مراجعه کنید.

تشخیص Haar-cascade در OpenCV

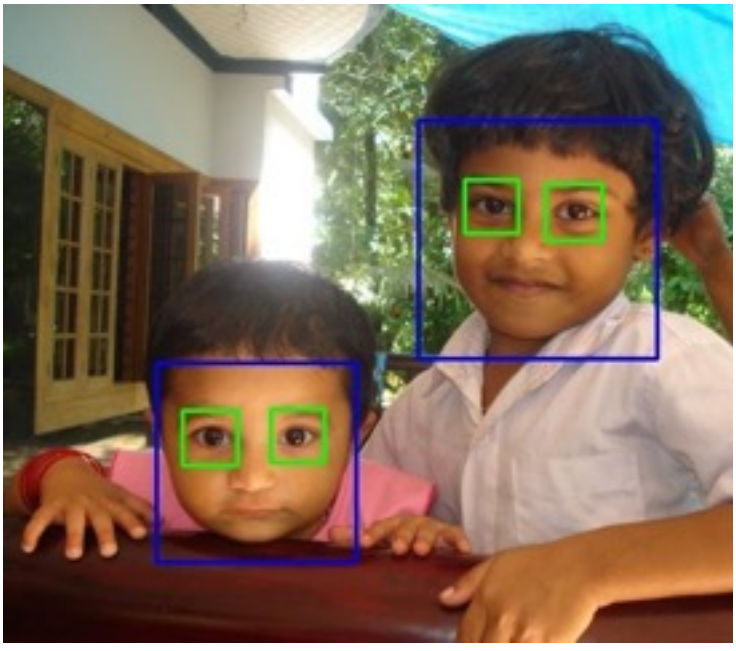
OpenCV آماده است تا شما در هر موردی به آن تمرین دهید، به عنوان مثال میتوانید کلاس خود را برای شناخت اشیایی مانند ماشین، هواپیما و غیره را بسازید. در این قسمت با یکسری کلاس که در

OpenCV برای تشخیص چهره و چشم موجود است آشنا میشویم که فایل های XML آن ها در پوشه /opencv/data/haarcascades ذخیره شده است.
برای این تمرین ابتدا باید فایل های XML مربوط به کاری که میخواهیم انجام دهیم را فراخوانی کنیم و سپس تصاویر را در حالت grayscale به برنامه بدهیم.

```
import numpy as np
import cv2
face_cascade =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
img = cv2.imread('sachin.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

حالا باید چهره را پیدا کنیم و بر اساس مختصاتی که داریم دور آن مربع رسم کنیم.

```
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
   cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
   roi_gray = gray[y:y+h, x:x+w]
   roi_color = img[y:y+h, x:x+w]
   eyes = eye_cascade.detectMultiScale(roi_gray)
   for (ex,ey,ew,eh) in eyes:
       cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),
2)
cv2.imshow('img',img)
```



منابع بیشتر برای یادگیری

برای مطالعه پیرامون این موضوع میتوانید به وبسایت رسمی OpenCV به آدرس opencv.org بروید و در قسمت مستندات علاوه بر توضیحات کامل پیرامون هر موضوع و الگوریتم، میتوانید نمونه کد ها را به زبان های برنامه نویسی مختلف را ببینید.
همچنین برای مطالعه بیشتر میتوانید کتاب های زیر را مطالعه کنید.

- Learning OpenCV
- OpenCV By Example
- Android Application Programming with OpenCV 3
- Learning Image Processing with OpenCV
- Mastering OpenCV with Practical Computer Vision Projects
- OpenCV 3.0 Computer Vision with Java
- OpenCV Computer Vision with Python
- OpenCV for Secret Agents
- Programming Computer Vision with Python

منابع

کتاب OpenCV By Example
کتاب Learning OpenCV
مستندات آموزش OpenCV با پایتون در http://opencv.org