گزارش اول

مقدمه

یکی از مشکلات توسعه دهندگان نرم افزار های بصری انجام محاسبات مختلف بر روی تصاویر است. به دلیل حجم بالای اطلاعات، پردازش آنها نیاز به برنامه نویسی بهینه دارد. شرکت Intel پروژه ای را با نام OpenCV آغاز کرد تا کتابخانه های بهینه برای انجام بلادرنگ محاسبات بصری، جهت توسعه دهندگان این دسته نرم افزار ها فراهم آورد.

بررسی opencv

OpenCV رایج ترین کتابخانه دید رایانه ای در دنیا با 14 میلیون بار دانلود است. ویژگی های زیر در این کتابخانه آن را به یک انتخاب عالی برای ساخت برنامه های تجاری Computer Vision تبدیل می کند. -بهینه سازی شده: OpenCV در C / C ++ با هدف ساخت برنامه های کاربردی در زمان واقعی نوشته شده است. این بسیار بهینه شده است هنگامی که با گزینه های مناسب کامپایل شده و می تواند هسته های متعدد را بر روی دستگاه خود استفاده کنید. همچنین با پشتیبانی از OpenCL با استفاده از منابع محاسباتی ناهمگن (مانند یک GPU) قادر به کار است. - دسترسی سطح باز (open source)تحت مجوز BSD: منبع باز و با مجوز تحت مجوز BSD مجاز است. این بدان معنی است که شما می توانید از آن برای ساخت برنامه های تجاری استفاده کنید و نیازی نیست کد منبع خود را باز کنید. با این حال، قسمت هایی از OpenCV وجود دارد (به عنوان مثال ماژول opencv\_contrib) که ممکن است تحت مجوز BSD باشد و یا تحت آن نباشد. - پیوند زبانی با دیگر زبان های برنامه نویسی: در واقع این کتابخانه با زبان c/c++ نوشته شده اما قابلیت تعامل با دیگر زبان های برنامه نویسی همچون python و java را نیز دارد. - قابلیت پشتیبانی سیستم عامل های متعدد: پشتیبانی از سیستم عامل های Linux، Mac، Windows، iOS و Android.

در این گزارش قصد داریم در چند بخش توابع و api های مربوط به opencv را مرور و بخش های کلی آن را با مثال بررسی کنیم. برای اینکار بر روی تصویر تمرکز کرده و عملیاتی هایی که میتوان اعمال کرد را بررسی میکنیم:

1. نحوه خواندن تصویر تغییر ابعاد و ذخیره کردن تصویر را مشاهده میکنیم:

با کمک تابع زیر میتوان تصاویر را خواند و بر روی ram آورد.

cv.imread("Hello.jpg")

با کمک تابع زیر می­توان ابعاد تصویر را کاهش داد:

img = scale(cv.imread("Hello.jpg"),coefficient=0.5)

که تابع scale به شکل زیر نوشته شده که ابعاد تصویر را خوانده و با کمک interpolation (میانگین گیری و برون یابی) سعی بر تعیین نقاط جدید دارد.

def scale(img,coefficient):

height, width, \_ = img.shape

return cv.resize(img,(int(coefficient\*width), int(coefficient\*height)),interpolation=cv.INTER\_AREA)

و با کمک تابع زیر میتوان تصویر را ذخیره نمود:

cv.imwrite("hey.jpg", img)

1. رسم خطوط بر روی تصویر مشاهده میکنیم:

با کمک تابع زیر میتوان یک مربع که محل نقطه اولیه آن، قطر مخالف آن، رنگ آن و قطر خط رسم شده و نحوه خط(خط کامل یا نقطه­چین) را مشخص نمود.

cv.rectangle(img, (20, 20), (300, 300), (255, 255, 0), thickness=2, lineType=cv.LINE\_8)

با کمک تابع زیر میتوان نحوه کشیدن یک خط را که پارامتر اول آن نقطه اول و پارامتر دوم آن مختصات دوم و پارامتر سوم رنگ و پارامتر آخر شیوه رسم خط را مشخص میکند، بررسی نمود.

cv.line(img, (20, 140), (380, 200), (0, 255, 0), thickness=4, lineType=cv.LINE\_AA)

با کمک تابع زیر میتوان نحوه یاداشت یک نوشته بر روی تصویر را انجام داد که در پارامتر اول متن نوشته و در پارامتر دوم مختصات نوشته و پارامترهای بعدی به ترتیب فونت و قطر خطوط ،رنگ و نحوه خطوط را مشخص میکند.

cv.putText(img, "this is Hossein :))))))))))", (90, 50), cv.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 255), 2)

با کمک تابع زیر میتوان یک دایره رسم نمود که پارمتر های آن به ترتیب مرکز دایره شعاع دایره، رنگ دایره قطر خطوط و نحوه خطوط را مشخص می­کند.

cv.circle(img, (200, 200), 150, (0, 0, 0), thickness=5, lineType=cv.LINE\_AA)



1. اعمال پردازش ها و transformation ها بر روی تصاویر

ایجاد یک ماتریس انتقال و ضرب در تصویر برای انتقال تصویر که با کمک تابع نوشته شده زیر انجام خواهد شد، با کمک این تابع میتوان تصویر واقعی را در صفحه به بخشی خاص منتقل کرد.

def change\_position\_to(img,x,y):

M = np.float32([ [1,0,x], [0,1,y]])

height, width, \_ = img.shape

return cv.warpAffine(img,M,(width,height))



ایجاد یک ماترسی دوران و ضرب در تصویر برای چرخاندن تصویر در فریم مشخص شده که با کمک تابع زیر انجام خواهد شد:

def rotaion(img,angel):

height, width, \_ = img.shape

M = cv.getRotationMatrix2D(((width-1)/2.0,(height-1)/2.0),angel,1)

return cv.warpAffine(img,M,(width,height))



ایجاد تبدیل affine transformation که تصویر را اسکیل و دوران میدهد. که توسط تابع زیر بدست می­آید.

def affine\_transformation(img,pst1,pst2):

rows,cols,ch = img.shape

M = cv.getAffineTransform(pts1,pts2)

return cv.warpAffine(img,M,(cols,rows))



زوم و پرسپکتیو کردن بر روی بخش مشخصی از تصویر و تغییر ابعاد تصویر که با کمک تابع زیر پیاده سازی شده:[[1]](#footnote-1)

def Perspective\_Transformation(img,pts1,x,y):

pts2 = np.float32([[0,0],[x,0],[0,y],[x,y]])

M = cv.getPerspectiveTransform(pts1,pts2)

return cv.warpPerspective(img,M,(x,y))

pts1 = np.float32([[516,286],[788,303],[541,656],[783,651]])

res6=Perspective\_Transformation(img, pts1=pts1, x=500,y=500)



1. از این تبدیل برای بزرگ کردن و مستقیم کردن پلاک در برنامه پلاک خوان میتوان استفاده نمود [↑](#footnote-ref-1)