Open CV使用总结

# 背景

Opencv是一个开源的计算机视觉处理库，采用C/C++编写，可以运行在多个操作系统上，具有丰富的使用接口。

本文档总结opencv的基本有用法，用于学习（代码实现基于python代码实现）。

# 基本操作

## 图片文件的读取和显示

读取图片文件并在窗口中呈现读取到的图片。

import cv2 as cv  
import numpy as np  
  
  
def load\_img\_frm\_file(fname):  
 img = cv.imread(fname)  
 if img is None:  
 print("read image file from file error")  
 return  
 cv.imshow("test\_wd", img)  
 cv.waitKey(1000)  
 cv.destroyAllWindows()  
  
  
load\_img\_frm\_file("E:\mlDataSet\cvdata\cat.jpg")  
load\_img\_frm\_file("E:\\mlDataSet\\cvdata\\timg.jpg")

## 将图片写入文件中

def write\_img\_to\_file(fname):  
 # load an image from file first and then write this image to a new file  
 img = cv.imread(fname)  
 if img is None:  
 print("read image file from file error")  
 return  
 cv.imshow("test\_wd", img)  
 cv.waitKey(1000)  
 cv.imwrite("E:\\mlDataSet\\cvdata\\test\_new.png", img)  
 cv.destroyAllWindows()  
  
  
write\_img\_to\_file("E:\\mlDataSet\\cvdata\\bird.jpg")

## 摄像头操作

Opencv可以用来操作摄像头，本节描述如何打开摄像设备并保存图像内容

def take\_picture\_from\_video\_capture():  
 cap = cv.VideoCapture(0) #parameter 0 describe this is the first video capturer  
 if not cap.isOpened():  
 print("open video capturer error")  
 return  
 while True:  
 ret, frm = cap.read()  
 if not ret:  
 return  
 #img = cv.cvtColor(frm, cv.COLOR\_BGR2GRAY)  
 img = frm  
 cv.imshow("tst\_wd", img)  
 cv.waitKey(10000)  
 cv.imwrite("E:\\mlDataSet\\cvdata\\test\_new.png", img)  
 break  
 cap.release()  
  
  
take\_picture\_from\_video\_capture()

下面这个例子从摄像头采集图像并持续显示

def take\_picture\_from\_video\_capture():  
 cap = cv.VideoCapture(0) #parameter 0 describe this is the first video capturer  
 if not cap.isOpened():  
 print("open video capturer error")  
 return  
 while True:  
 ret, frm = cap.read()  
 if not ret:  
 return  
 img = cv.cvtColor(frm, cv.COLOR\_BGR2GRAY)  
 cv.imshow("tst\_wd", img)  
 if cv.waitKey(1) == ord('q'):  
 break  
 cap.release()  
  
  
take\_picture\_from\_video\_capture()

从摄像头上采集到的数据，还可以写入到文件中，如下代码描述如何将摄像头采集到的视频写入文件

def save\_video\_to\_file\_for\_video\_capture():  
 cap = cv.VideoCapture(0)  
 if not cap.isOpened():  
 print("open video capturer error")  
 return  
  
 # define code and decode object  
 fourcc = cv.VideoWriter\_fourcc(\*'XVID')  
 out = cv.VideoWriter("E:\\mlDataSet\\cvdata\\output.avi", fourcc, 20.0, (640, 480))  
 while True:  
 ret, frame = cap.read()  
 if not ret:  
 break  
 #frame = cv.flip(frame, 0)  
 out.write(frame)  
 cv.imshow("tst\_wd", frame)  
 if cv.waitKey(1) == ord('q'):  
 break  
 cap.release()  
 out.release()  
 cv.destroyAllWindows()  
  
  
save\_video\_to\_file\_for\_video\_capture()

# Opencv绘图函数操作

使用opencv也可以完成基本图型的绘制，例如画线，画圆等。

def draw\_basic\_picture\_with\_draw\_function():  
 img = np.zeros((512, 512, 3), np.uint8)  
 cv.line(img, (0, 0), (511, 511), (255, 0, 0), 5)  
 cv.rectangle(img, (384, 0), (510, 128), (0, 255, 0), 3)  
 cv.circle(img, (447, 63), 63, (0, 0, 255), -1)  
 cv.ellipse(img, (256, 256), (100, 50), 0, 0, 180, 255, -1)  
  
 font = cv.FONT\_HERSHEY\_SIMPLEX  
 cv.putText(img, 'OpenCV', (10, 500), font, 4, (255, 255, 255), 2, cv.LINE\_AA)  
 cv.imshow("tst", img)  
 cv.waitKey(0)  
  
  
draw\_basic\_picture\_with\_draw\_function()

# Opencv采集鼠标或键盘事件并作处理

如下例子，我们响应鼠标事件，并在窗口中进行图型绘制

def handle\_mouse\_event():  
 img = np.zeros((512, 512, 3), np.uint8)  
  
 def draw\_circle\_with\_mouse\_event(event, x, y, flags, param):  
 if event == cv.EVENT\_LBUTTONDBLCLK:  
 cv.circle(img, (x, y), 100, (255, 0, 0), -1)  
 return  
  
 cv.namedWindow('image')  
 cv.setMouseCallback('image', draw\_circle\_with\_mouse\_event)  
  
 while True:  
 cv.imshow('image', img)  
 if cv.waitKey(20) & 0xff == 27:  
 break  
 cv.destroyAllWindows()  
  
  
handle\_mouse\_event()

# 图像基本操作

本部分主要描述对图像的基本操作，比如说对图像的翻转，图像像素的修改等

## 图像的几何操作

### 对图像进行扩大或缩放

图像的扩大或缩小，可以使用cv.resize接口实现，样例代码如下：

def image\_scalinig\_operation():  
 img = cv.imread('002.jpg')  
 cv.imshow('test', img)  
 dst\_img = cv.resize(img, None, fx=0.5, fy=0.5, interpolation=cv.INTER\_CUBIC)  
 cv.imshow('test2', dst\_img)  
  
 cv.waitKey(0)  
 cv.destroyAllWindows()

### 图像的平移操作

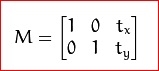
Opencv中通过仿射变换函数warpAffine()进行图像的平移，反转等操作。

仿射变换是二维坐标之间的线性变换，故变换后的图像仍具有原始图像的一些特性。

warpAffine()函数通过一个预先定义好的矩阵M对图像原始数据进行计算，然后得到最终结果，M是一个2\*3的衍射矩阵。

假定M矩阵的取值内容如下：

对于图像的平移，该矩阵可以定义为如下



的样子。

## 图像轮廓识别

可以简单的将连续的具有相同颜色特征的一系列点看做是图像的轮廓，通过轮廓识别可以检查图像中的物体，形状等。

在opencv中通过接口函数[cv.findContours](https://docs.opencv.org/4.1.0/d3/dc0/group__imgproc__shape.html" \l "gae4156f04053c44f886e387cff0ef6e08)识别对应的轮廓数据；通过接口函数cv.drawContours将轮廓绘制在原始图像上。

具体使用方式如下：

def contours\_find\_operation():  
 img = cv.imread("yanjiuling\_002.jpg")  
 grayimg = cv.cvtColor(img, cv.COLOR\_BGR2GRAY)  
 ret, thresh = cv.threshold(grayimg, 127, 255, 0)  
 contours, hierarchy = cv.findContours(thresh, cv.RETR\_TREE, cv.CHAIN\_APPROX\_SIMPLE)  
 cv.drawContours(img, contours, -1, (0, 255, 0), 2)  
 cv.imshow("img", img)  
 cv.waitKey(0)  
 cv.destroyAllWindows()

### Moment 图像的矩

一个从数字图形中计算出来的矩集，通常描述了该图像的全局特征，并提供了大量的关于该图像不同类型的几何特征信息，比如大小、位置、方向及形状等。

一阶矩与形状有关，二阶矩显示了曲线围绕直线平均值的扩展程度，三阶矩则是关于平均值的对称性测量。

**不变矩：**由二阶矩和三阶矩可以导出一组共七个不变矩。不变矩是图像的统计特征，满足平移、伸缩、旋转均不变的不变形，在图像处理中，几何不变矩可以作为一个重要的特征来表示物体，可以据此特征来对图像进行分类等操作。

对于图像（单通道图像）来说，图像可以看成是一个平板物体，其一阶矩和零阶矩就可以拿来计算某个形状的重心，而二阶矩就可以拿来计算形状的方向。

图像的矩可以帮助计算图像中物体的一些特征，例如中心点等。

### 图像Convex Hull（ 凸包）

凸包（Convex Hull）是一个计算几何（图形学）中的概念。

在一个实数向量空间V中，对于给定集合X，所有包含X的凸集的交集S被称为X的凸包。X的凸包可以用X内所有点(X1，...Xn)的凸组合来构造.

在二维欧几里得空间中，凸包可想象为一条刚好包著所有点的橡皮圈。

用不严谨的话来讲，给定二维平面上的点集，凸包就是将最外层的点连接起来构成的凸多边形，它能包含点集中所有的点。

凸包最常用的凸包算法是Graham扫描法和Jarvis步进法

### 图像的Bounding Rectangle（外接矩形）

### 图像中的Minimum Enclosing Circle（最小闭合圆）

## 轮廓识别中的关键特征

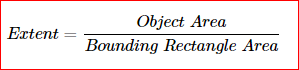
### Aspect Ratio纵横比

一个矩形物体宽和高的比例



### Extent

图像中对象区域与边界矩形区域的比例



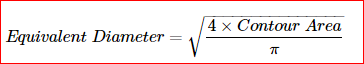
### Solidity

图像密实度指图像中物体面积与其凸包面积比例



### Equivalent Diameter等价直径

Equivalent Diameter is the diameter of the circle whose area is same as the contour area.



### Orientation

图像中一个物体朝向某个方向的角度

### 掩码和像素点Mask and Pixel Points

有些场景下，需要图像中某个物体包括的所有点信息

### Mean Color or Mean Intensity

均值颜色以及均值烈度

在灰度图像模式下，可以找到某个对象物体的平均颜色取值和密度取值

### Extreme Points

超级点一般指图像中某个物体，最上，最下，最左，最右的点。

## 图像直方图处理

图像直方图反应了一幅图像中像素的分布，横坐标代表了图像像素的种类，纵坐标代表了图像像素的取值，在实际工程中，直方图在特征匹配，特征提取方面有很好的应用。

### 直方图数据获取

### 直方图均衡化

### 2维直方图

### 直方图反向投影

如果一幅图像区域中显示的是一种结构纹理或一个独特的物体，那么这个区域的直方图可以看作一个概率函数，该函数描述的是某个像素属于该纹理或物体的概率。

所谓反向投影，就是先计算某一特征的直方图模型，然后用该模型寻找测试图像中存在的特征。

## 图像的Fourier Transform（傅里叶变换）

 2D Discrete Fourier Transform (DFT)离散傅里叶变化可以被用来发现图像的频率域，一个快速傅里叶变换函数被应用到DFT上。

傅里叶理论讲的是：任何信号都可以标识成一系列正弦信号的叠加，在图像领域中就是将图像的“亮度”作为正弦变量。

图像的频率是表征图像中灰度变化剧烈程度的指标，是灰度在平面空间上的梯度，在噪声点和图像边缘处的频率为高频。

从物理效果上看，傅里叶变化就是将图像从空间域转换为频率域，其逆变换就是将图像从频率率转换到空间域。在傅里叶变换前，图像是由对在连续空间上的采样得到的一系列点的集合，习惯用一个二维矩阵标识，图像可以由z=f(x,y)表示；由于空间是三维的，图像是二维的，因此空间中物体在另外一个维度上的关系就由梯度来表示，这样可以通过观察图像信息得知物体在三维空间中的对应关系。实际上对图像进行二维傅里叶变换得到的频谱图，就是图像梯度的分布图。

傅里叶频谱图上得到的明暗不一的点，实际上是图像上某一个点与其领域点差异的强弱，即梯度的大小，也即该点的频率，（可以这么理解，图像中的低频部分指低梯度的点，高频部分相反），一般来讲，梯度大则该点的亮度强，否则该点亮度弱。

这样通过观察傅里叶变换后的频谱图（功率图）可以得到图像的能量分布，如果频谱图中暗点多，则图像柔和，反之，则图像尖锐。

傅里叶变换进行图像处理由以下几个特点

1：直流成分F(0,0)等于图像的平均值；

2： 能量频谱|F（u,v）|^2完全对称于原点；其中F=PfQ, f表示原图，P和Q都是对称的实正交矩阵，这个公式表示傅里叶变换就是个正交矩阵的正交变换

3：图像f平移（a,b）后，F只有exp[-2pij(au/M+bv/M)]的相位变化，能量频谱不发生变化。

4：图像f自乘平均等于能量频谱的总和，f的分散等于能量频谱中除直流成分后的总和。

5：图像f(x,y)和g(x,y)的卷积h(x,y)=f(x,y)\*g(x,y)的傅里叶变换H（u，v）等于f(x,y)和g(x,y)各自的傅里叶变换的乘积。

图像中的每个点通过傅里叶变换都成了谐波函数的组合，也就由了频率，这个频率是在这一点上所产生的灰度频率之和，也就是说傅里叶变换可以将这些频率分开，当想除去图像背景时，只要去掉背景频率即可以。

二维图像进行傅里叶变换后，傅里叶图像的中间部分时图像的低频部分，外缘为高频部分；一般高频反映图像的细节，低频反映图像的轮廓概貌，因此在图像处理时，将高频部分设置为0，从而欺骗人眼。

## [Hough transform(霍夫变换)](https://www.cnblogs.com/AndyJee/p/3805594.html)

## 分水岭算法（Watershed algorithm）

## grabcut算法

# 一些有趣的实现样例

## 通过傅里叶变化在图片中添加盲水印信息