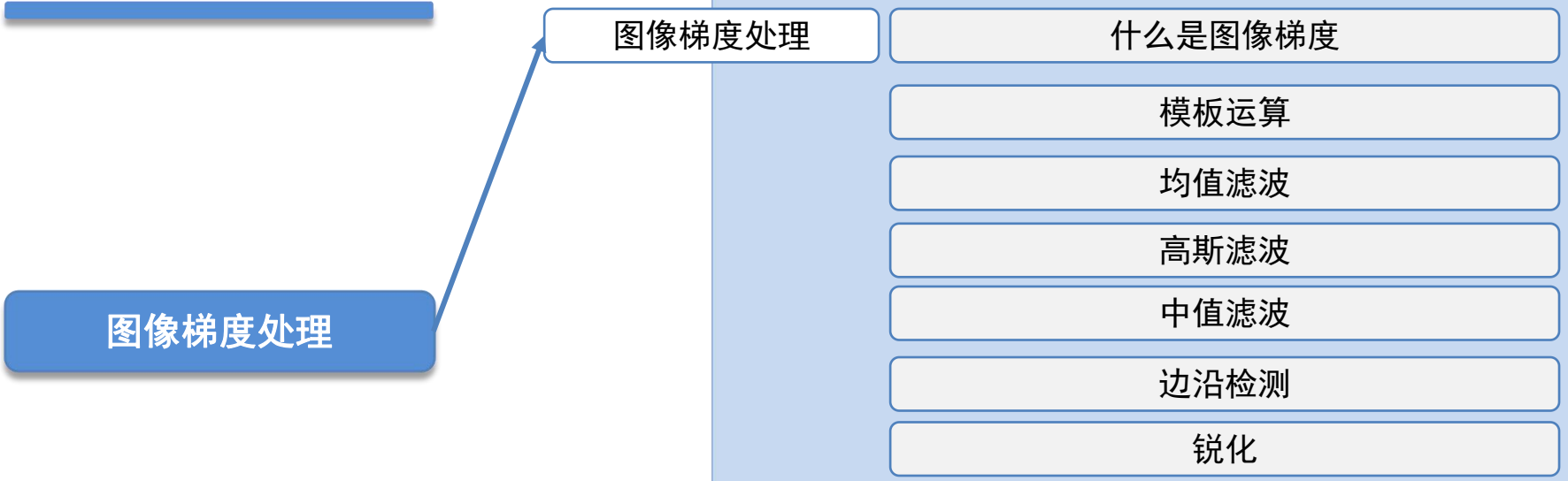
深度学习

计算机视觉基本理论

DAY03

图像梯度处理



图像梯度处理

什么是图像梯度

- 图像梯度计算的是图像变化的速度。对于图像的边缘部分，其灰度值变化较大，梯度值也较大；相反，对于图像中比较平滑的部分，其灰度值变化较小，相应的梯度值也较小。一般情况下，图像梯度计算的是图像的边缘信息。

知识讲解

模板运算

知识讲解

- 模板（滤波器）是一个尺寸为n*n的小图像W（n一般取奇数，称为模板尺寸），每个位置上的值w被称为权重。在进行计算时，将模板的中心和像素P对齐，选取原始图像中和模板相同范围的邻域N的像素值作为输入。
- 模板卷积的计算是将对齐后的对应位置像素相乘，再进行累加作为像素P位置的输出值。记原始图像的像素灰度值为s, 计算后的值为d, 则P点的输出值 $d = \frac{\sum w_i s_i}{\sum w_i}$
- 模板排序的计算时将邻域N的像素值进行排序，选择特定次序的灰度值，作为像素P位置的输出值，如最大值、最小值、中位数等。

均值滤波

- 均值滤波指模板权重都为1的滤波器。它将像素的邻域平均值作为输出结果，均值滤波可以起到图像平滑的效果，可以去除噪声，但随着模板尺寸的增加图像会变得更为模糊。经常被作为模糊化使用。

知识讲解

1	1	1
1	1	1
1	1	1

原图

3x3均值滤波

5x5均值滤波

高斯滤波

- 为了减少模板尺寸增加对图像的模糊化，可以使用高斯滤波器，高斯滤波的模板根据高斯分布来确定模板系数，接近中心的权重比边缘的大。5的高斯滤波器如下所示：

知识讲解

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

原图

5x5均值滤波

5x5高斯滤波

中值滤波

知识讲解

- 中值滤波属于模板排序运算的滤波器。中值滤波器将邻域内像素排序后的中位数值输出代替原像素值。它在实现降噪操作的同时，保留了原始图像的锐度，不会修改原始图像的灰度值。
- 中值滤波的使用非常普遍，它对椒盐噪声的抑制效果很好，在抑制随机噪声的同时能有效保护边缘少受模糊。但中值滤波是一种非线性变化，它可能会破坏图像中线性关系，对于点、线等细节较多的图像和高精度的图像处理任务中并不太合适。

原图

3x3均值滤波

3x3中值滤波

边沿检测

- 通过梯度计算可以获取图像中细节的边缘。为在锐化边缘的同时减少噪声的影响，通过改进梯度法发展出了不同的边缘检测算子：
  - ✓ 一阶梯度：Prewitt梯度算子、Sobel梯度算子
  - ✓ 二阶梯度：Laplacian梯度算子。

知识讲解

-1	0	1
-1	0	1
-1	0	1

-1	-1	-1
0	0	0
1	1	1

Prewitt算子

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

Sobel算子

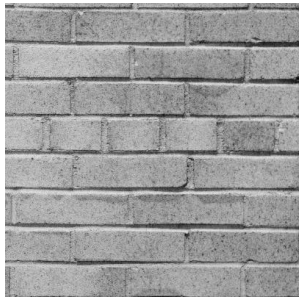
0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

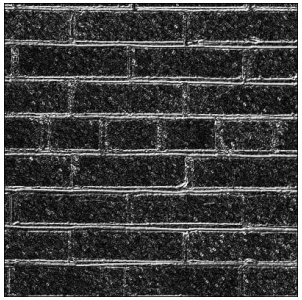
Laplacian算子

边沿检测（续）

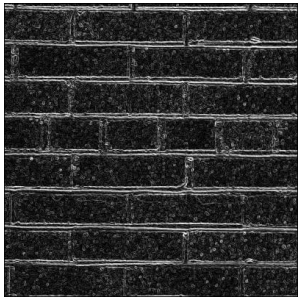
- 边沿检测效果



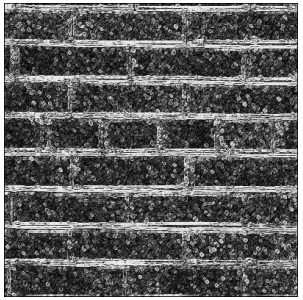
原图



Prewitt算子



Sobel算子



Laplacian算子

知识讲解

锐化

- 图像锐化与图像平滑是相反的操作，锐化是通过增强高频分量来减少图像中的模糊，增强图像细节边缘和轮廓，增强灰度反差，便于后期对目标的识别和处理。锐化处理在增强图像边缘的同时也增加了图像的噪声。

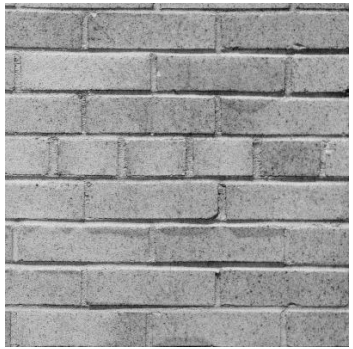
知识讲解

锐化（续）

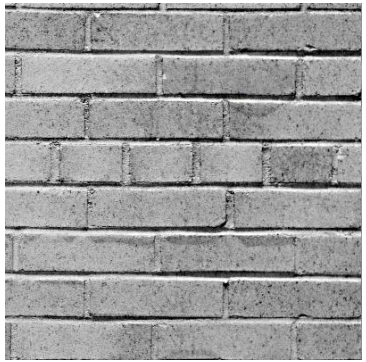
- 将求取的边缘按照一定系数比例叠加到原始图像上，即可实现对图像的锐化操作。例如使用Laplacian梯度算子进行锐化操作的模板，其中A是大于等于1的系数：

知识讲解

0	-1	0
-1	A+4	-1
0	-1	0



原图



Laplacian锐化后的效果

图像轮廓

图像轮廓

- 什么是图像轮廓
- 查找和绘制轮廓
- 轮廓拟合
- 最小包围圆形
- 最优拟合椭圆
- 逼近多边形

图像轮廓

图像轮廓

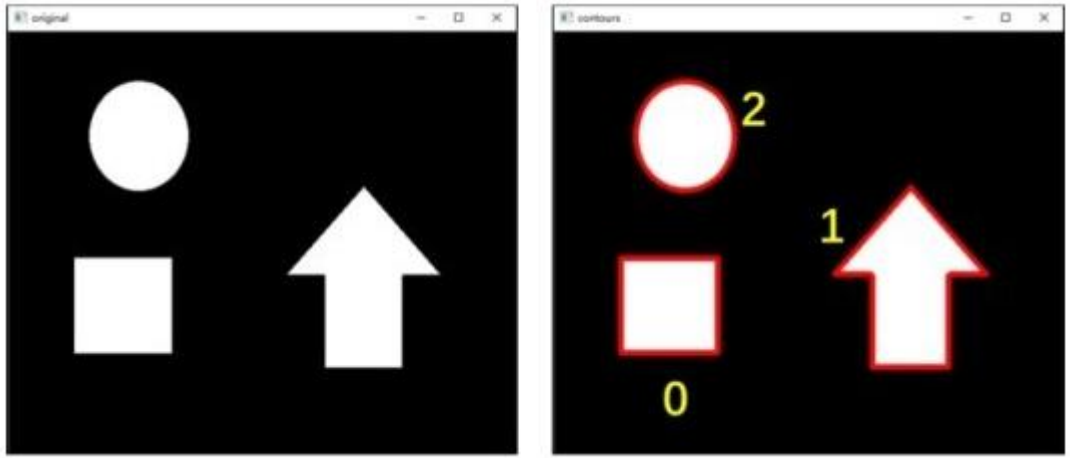
什么是图像轮廓

- 边缘检测虽然能够检测出边缘，但边缘是不连续的，检测到的边缘并不是一个整体。图像轮廓是指将边缘连接起来形成的一个整体，用于后续的计算。
- 图像轮廓是图像中非常重要的一个特征信息，通过对图像轮廓的操作，我们能够获取目标图像的大小、位置、方向等信息。
- 图像轮廓操作包括：查找轮廓、绘制轮廓、轮廓拟合等

知识讲解

查找和绘制轮廓

- 一个轮廓对应着一系列的点，这些点以某种方式表示图像中的一条曲线，将这些点绘制成不同样式的线条，就是轮廓查找与绘制



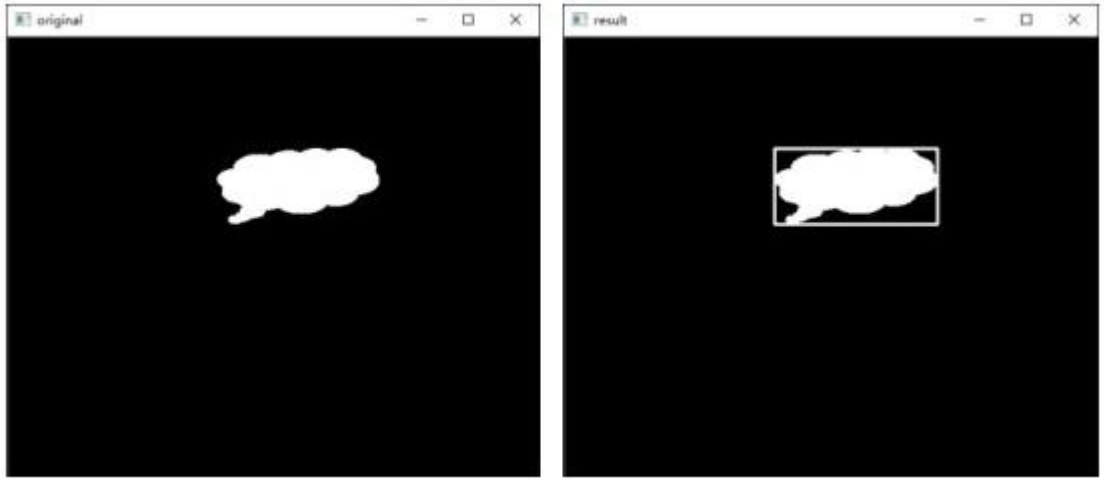
知识讲解

轮廓拟合

- 在计算轮廓时，可能并不需要实际的轮廓，而仅需要一个接近于轮廓的近似多边形，绘制这个近似多边形称之为轮廓拟合

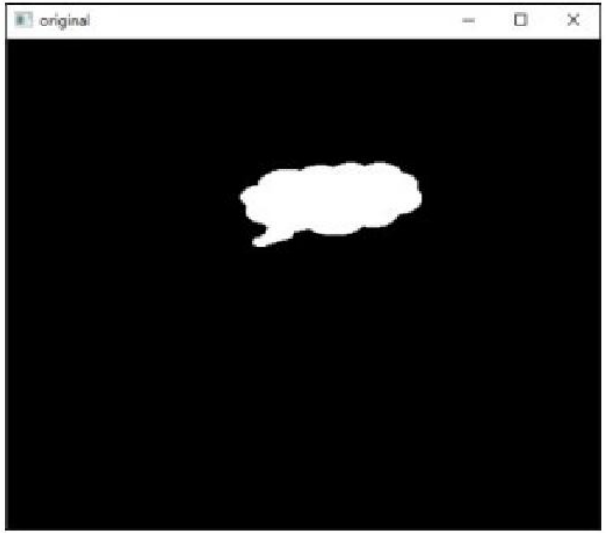
知识讲解

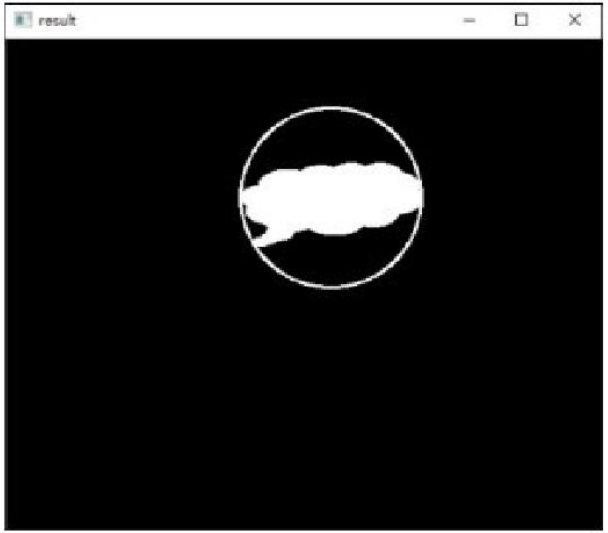
矩形包围框



知识讲解

最小包围圆形





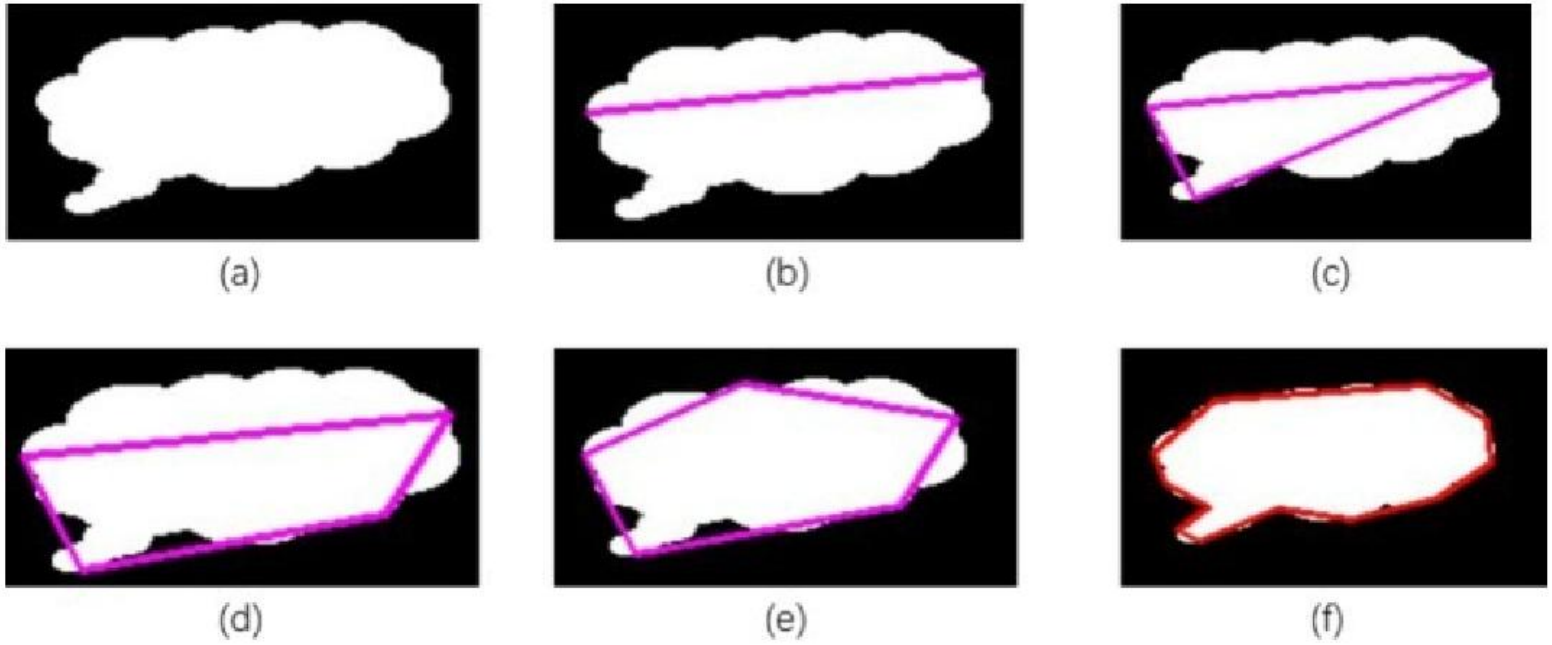
知识讲解

最优拟合椭圆

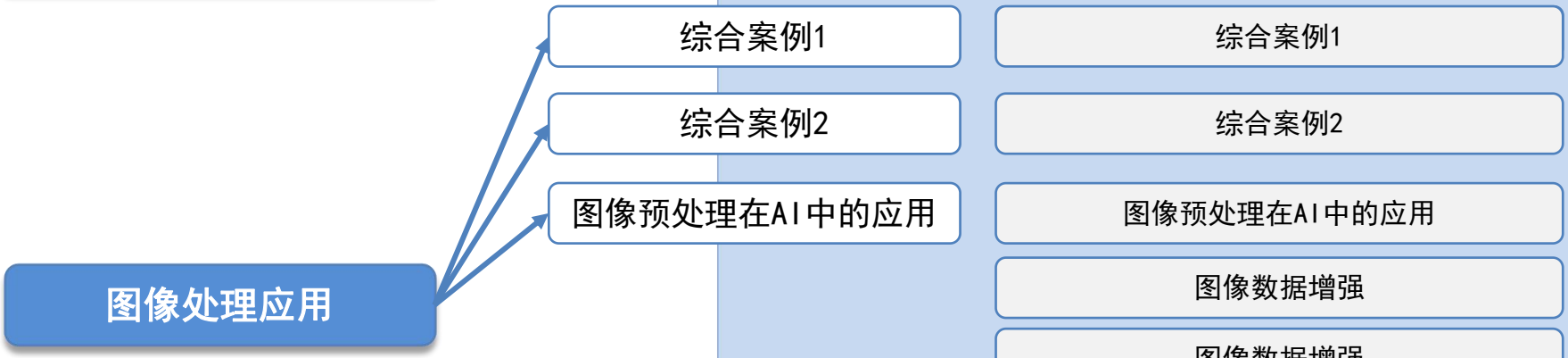
知识讲解

逼近多边形

知识讲解



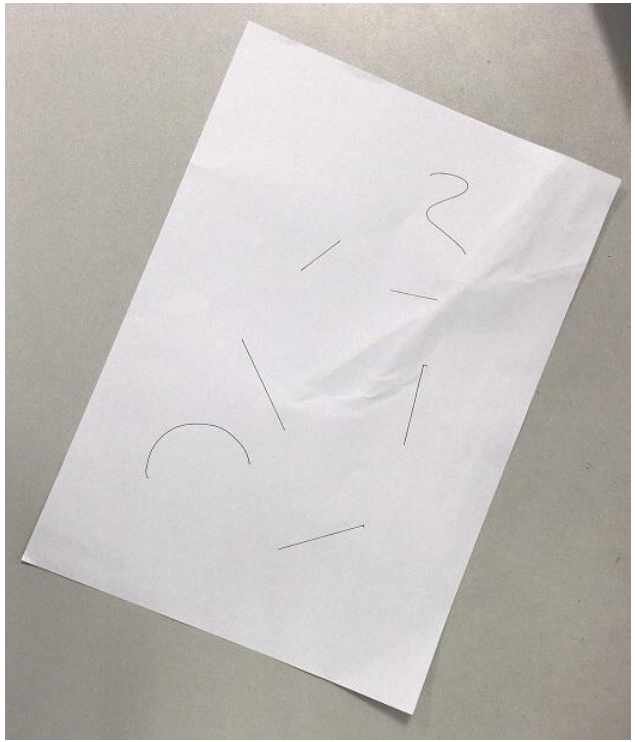
图像处理应用


图像处理应用
综合案例1
综合案例2
图像预处理在AI中的应用
综合案例1
综合案例2
图像预处理在AI中的应用
图像数据增强
图像数据增强
纯图像技术的缺陷

综合案例

综合案例

- 任务描述：我们对图像中的目标进行分析和检测时，目标往往具有一定的倾斜角度，自然条件下拍摄的图像，完全平正是很少的。因此，需要将倾斜的目标“扶正”的过程就就叫做图像矫正。该案例中使用的原始图像如右图所示。



课堂练习

综合案例（续一）

课堂练习

```
# 图像校正示例
import cv2
import imutils
import numpy as np

im = cv2.imread("../data/paper.jpg")
gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
cv2.imshow('im', im)

# 模糊
blurred = cv2.GaussianBlur(gray, (5, 5), 0)
# 膨胀
dilate = cv2.dilate(blurred,
                    cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3)))
# 检测边沿
edged = cv2.Canny(dilate,  # 原始图像
                  30, 120,  # 滞后阈值、模糊度
                  3)  # 孔径大小
```

综合案例（续二）

课堂练习

```
19 # 轮廓检测
20 cnts = cv2.findContours(edged.copy(),
21                         cv2.RETR_EXTERNAL,  # 只检测外轮廓
22                         cv2.CHAIN_APPROX_SIMPLE)  # 只保留该方向的终点坐标
23 cnts = cnts[0] if imutils.is_cv2() else cnts[1]  # 判断是opencv2还是opencv3
24 docCnt = None
25
26 # 绘制轮廓
27 # im_cnt = cv2.drawContours(im,  # 绘制图像
28 #                           cnts,  # 轮廓点列表
29 #                           -1,  # 绘制全部轮廓
30 #                           (0, 0, 255),  # 轮廓颜色：红色
31 #                           2)  # 轮廓粗细
32 # cv2.imshow("im_cnt", im_cnt)
```

综合案例（续三）

```
34 # 计算轮廓面积，并排序
35 if len(cnts) > 0:
36     cnts = sorted(cnts,  # 数据
37                   key=cv2.contourArea,  # 排序依据，根据contourArea函数结果排序
38                   reverse=True)
39     for c in cnts:
40         peri = cv2.arcLength(c, True)  # 计算轮廓周长
41         approx = cv2.approxPolyDP(c, 0.02 * peri, True)  # 轮廓多边形拟合
42         # 轮廓为4个点表示找到纸张
43         if len(approx) == 4:
44             docCnt = approx
45             break
46
47 print(docCnt)
```

课堂练习

综合案例（续四）

课堂练习

```
# 用圆圈标记处角点
points = []
for peak in docCnt:
    peak = peak[0]
    # 绘制圆
    cv2.circle(im,  # 绘制图像
               tuple(peak), 10,  # 圆心、半径
               (0, 0, 255), 2)  # 颜色、粗细
    points.append(peak)  # 添加到列表
print(points)
```

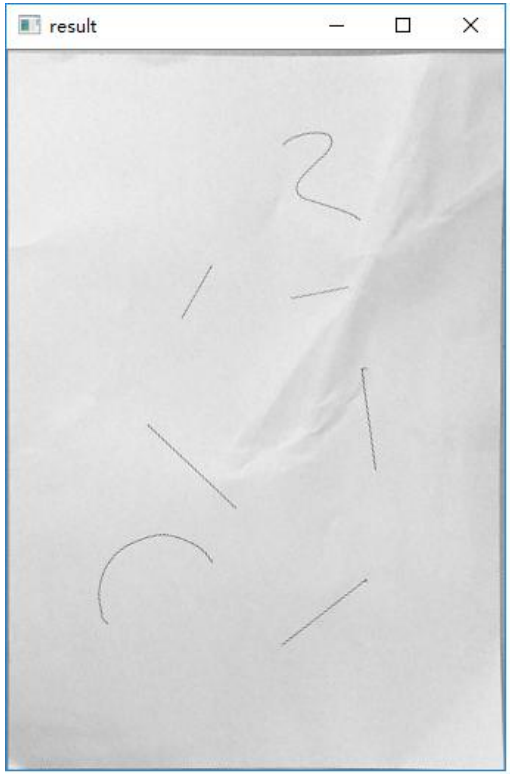
综合案例（续五）

课堂练习

```
60 # 校正
61 src = np.float32([points[0], points[1], points[2], points[3]])  # 原来逆时针方向四个点
62 dst = np.float32([[0, 0], [0, 488], [337, 488], [337, 0]])  # 对应变换后逆时针方向四个点
63 m = cv2.getPerspectiveTransform(src, dst)  # 生成透视变换矩阵
64 result = cv2.warpPerspective(gray.copy(), m, (337, 488))  # 透视变换
65 cv2.imshow("result", result)  # 显示透视变换结果
66
67 cv2.waitKey()
68 cv2.destroyAllWindows()
```

综合案例（续六）

- 校正效果



课堂练习

图像预处理在AI中的应用

图像预处理在AI中的应用

- 图像预处理的目的，是让图像数据更适合AI模型进行处理，例如调整大小、颜色
- 通过图像预处理技术，实现数据集的扩充，这种方法称为数据增强。数据增强主要方法有：缩放，拉伸，加入噪点，翻转，旋转，平移，剪切，对比度调整，通道变化。

知识讲解

图像数据增强

知识讲解

原图

通道调整

水平翻转

缩放

拉伸

旋转

噪声

裁剪

纯图像技术的缺陷

- 到目前为止，我们使用的基本是纯图像技术，对图像大小、颜色、形状、轮廓、边沿进行变换和处理，但这些技术都有一个共同的缺点，即无法理解图像内容和场景，要实现这个目标，必须借助于深度学习技术。

知识讲解

今日总结

- 图像形态处理
- 图像轮廓
- 计算机图像技术应用