Assignment3

视觉组考核(OpenCV)

特征与目标提取

什么是特征？什么是目标提取？在Assignment1中的色彩分割任务中我们已经接触到了这些内容。无论是棕色叉还是绿色钩，颜色和几何形状都是他们的特征，而需要做的就是依靠这些**独占性**的特征将他们和其他图形区分开来。在条件允许的情况下，我们倾向于选择最方便快捷的方法，例如颜色；但若图中还有其他的绿色形状，在提取出绿色钩时就必须再借助它的几何形状特征。

为了在图像处理中**侦测和量化几何特征**以及相应的**可视化需求**，OpenCV提供了一系列API。在学习他们之前，必须首先意识到这些API并非无所不能，他们的功能往往依然隶属于**预处理阶段**，更加关键的筛选或分类等**逻辑过程依然需要我们自己实现。**
长篇累牍的逻辑判断会引入大量参数，使得代码不再优雅并且调试困难，例如有时新引入的判断条件 B 完全是某个已经存在的条件 A 的子集，即有 $P(B|A)=1$，而我们却对此一无所知
对特征的深入思考(或者说**更高层次的抽象**)有时可以避免这些问题，在**简洁和高效**上取得双赢。当然这对程序的书写者来说是一个不小的挑战。

在这个阶段我们往往讨论的是经过一些前期处理而得的**二值图像**(逻辑0和1，即零和非零)。
主要的分析手段：**Structural Analysis and Shape Descriptors**，涵盖了接下来将介绍的所有内容(除了Hough变换，Radon变换)

Contour Detection

Using contour detection, we can detect the borders of objects, and localize them easily in an image. It is often the first step for many interesting applications, such as image-foreground extraction, simple-image segmentation, detection and recognition. 【1】
你需要了解**Contour Detection** 和 **Edge Detection**的区别。
请学习：Contour Detection using OpenCV
请尝试：对**未二值化**的图像使用findContours()会如何？drawContours()使用在灰度图或二值图上会如何？

几何特征的量化

提取到的轮廓一般还需要继续处理，不过也有非常直接的应用情景——判断某个点是否在轮廓内：

pointPolygonTest()

就在如上函数下方列有一个重要的函数：**rotatedRectangleIntersection()**，你可以最后回过头来看一下

> 这里插一嘴，我们之所以知道这些函数并非我们全文背诵了OpenCV的官方文档。有时正是编程中的想法要求我们去查找是否有现成的API可以实现需求。因此一方面要积累经验，另一方面要注重效率，避免在开发过程中闭门造车。当然为了深入学习某些功能，适当地阅读源码或原始论文也是必要的

轮廓的基本属性

- 面积：**contourArea()** Tips：Green formula(格林公式)
- 周长： **arcLength()**

思考或实验：

- 创建一个正方形，将它的四个顶点**有序**装入轮廓并传入contourArea()，对于顺时针装入和逆时针装入，函数返回结果如何(`oriented`设置为`true`)？就面积而言，是否等于 $边长^2$ ？
- **contourArea()**返回的面积和**轮廓所包围的像素个数**和**数学公式计算的面积**分别如何？请向我们举例。你需要先比较自己设定的简单图形，例如矩形和圆；再比较实际工程中的不规则轮廓(不规则轮廓不需要数学计算)
- 对于`std::vector<cv::Point2f>{cv::Point2f(80, 80), cv::Point2f(80, 810)}`和`std::vector<cv::Point2f>{cv::Point2f(0, 0), cv::Point2f(80, 80)}`，**arcLength()**分别会给出什么结果？
- 对于一个std::vector对象表示的轮廓，比较**arcLength()**和成员函数**std::vector::size()**以及**数学公式计算**给出的结果(需要注意**CHAIN_APPROX_NONE**和**CHAIN_APPROX_SIMPLE**)。你需要先比较自己设定的简单图形，例如矩形和圆；再比较实际工程中的不规则轮廓(不规则轮廓不需要数学计算)
- 以上所需的不规则轮廓请以文末图片为例

- 现在我们对于面积和周长各有两种衡量方法：
  面积：**contourArea()**，**轮廓所包围的像素个数**
  周长：**arcLength()**，**轮廓的像素个数(`size()`)**
  对于简单图形(矩形、圆、三角形)而言，两种衡量方法中谁给出的结果会更接近数学计算？
  对于面积和周长，我们有时需要他们之间的关系符合数学(以圆为例，$S=\pi\left(\frac{L}{2\pi}\right)^2$)，而不要求单一结果的数值精确。现有四种组合方式：
  (**contourArea()**，**arcLength()**)， …… ，(**轮廓所包围的像素个数**，**轮廓的像素个数(`size()`)**)，对半径分别为10，50，100的圆，比较以上四种组合方式

轮廓拟合

矩形：**boundingRect()**，**minAreaRect()**
圆：**minEnclosingCircle()**：Finds a circle of the minimum area enclosing a 2D point set
直线：**fitLine()**
椭圆：自行了解

你可能会思考，是否可以求**最大内接矩形**，**最大内接圆**？对于原始的轮廓来说，前者可能相当困难，不过存在一个退化的实现：Computing the largest orthogonal rectangle in a convex polygon；后者则可以利用**pointPolygonTest()**

多边形近似：**approxPolyDP()**：Approximates a polygonal curve with the specified precision
Convex Hull：很重要的概念，往往用于加强**approxPolyDP()**的结果。参考：
https://learnopencv.com/convex-hull-using-opencv-in-python-and-c/
https://brilliant.org/wiki/convex-hull/
convexHull()

形状匹配：**matchShapes()**：检测两个形状之间的相似度，返回值越小，越相似

可视化

OpenCV具有**Drawing Functions**
常用的有：**drawContours()**，**fillConvexPoly()**(不仅可以填充凸多边形，还可以填充任何没有自相交的单调多边形)，**circle()**，**rectangle()**，**line()**，**polylines()**
参考：**Annotating Images Using OpenCV**：https://learnopencv.com/annotating-images-using-opencv/

Hough变换

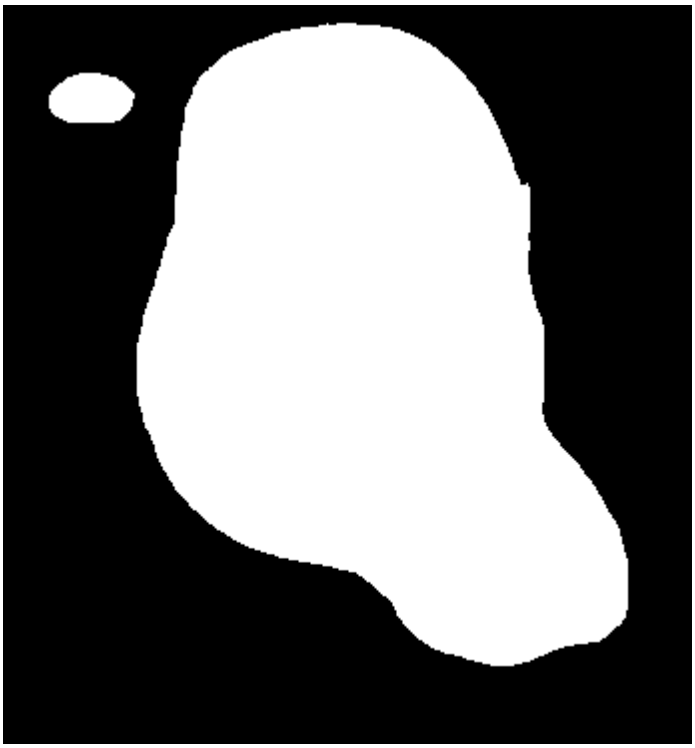
Hough变换和上述的轮廓拟合有一定区别，它可以自动检测图片中的结构。而轮廓拟合更倾向于使用我们主动提取好的数据去获取一个数学的描述或近似。Hough变换主要有以下内容：

- **Hough Line Transform**
- **Hough Circle Transform**
  思考：**minEnclosingCircle()**和**HoughCircles()**的抗噪声能力如何

Radon变换

从这篇文章你可以粗浅地了解Radon变换：https://zhuanlan.zhihu.com/p/61492627
Radon变换相较于上面的内容略为“小众”。了解这个概念，实际应用中它可能会成为你灵感的源泉。

不规则轮廓图源



请使用大的那个

提交要求

1. 一份学习笔记(PDF)
2. 学习笔记需要包含：
    - 简要总结新学习到的知识
    - **对本篇内容中问题的思考和必要的案例佐证，数据记录**
    - **心得体会**(对你感兴趣的内容的注解等或者其他值得记录的内容)
    - 对于本教程的建议(Optional)
3. 截止日期：**见群通知**
4. 邮件格式：
    - 邮件以 **视觉考核-姓名-年级** 的格式命名
5. 发送至正确的邮箱