第七讲 机器视觉 1

2025/05/10

文件：机器视觉-1.pdf、机器视觉-1-code.zip
作业：机器视觉-1

0. 机器视觉与图像处理

- 机器视觉是人工智能正在快速发展的一个分支。简单来说，机器视觉就使用机器**代替人眼**来做测量和判断
- 机器视觉是一项**综合技术**，一个典型的机器视觉应用系统包括图像捕捉、光源系统、图像数字化模块、数字图像处理模块、智能判断决策模块和机械控制执行模块
- 机器视觉系统是通过**机器视觉摄取装置**将被社区目标转换成图像信号，传达给专用的**图像处理系统**，得到被摄目标的形态信息，根据像素分布和亮度、颜色等信息，转变成数字化信号；图像系统对这些信号进行各种运算来抽取目标的特征，进而根据判别地接过来**控制现场的设备动作**

1. 图像是如何产生的

1.1 相机组件介绍

- **光圈**：作用类似瞳孔，通过调节光圈大小控制进入相机光线数量，光圈大小用 f 表示，$\mathrm{f}/1.4$ 的镜头比 $\mathrm{f}/2.8$ 的镜头更亮，以此类推
- **快门**：快门速度越慢，曝光时间越长，照片会越亮；快门速度还会影响在运动中的拍摄效果，如果快门速度不够快，会因运动导致物体上的同一点在多个像素上曝光而引起运动模糊
- **感光度**：是一个用来衡量数码相机中感光元件对光线的灵敏度的值，通常用 ISO 表示，ISO 越大，照片越亮，但噪点越多

1.2 相机传感器的工作原理

- 图像获取系统的组成永昌包括三个部分：**能量源**，**光学系统**和**图像传感器**
- **光学系统**是指由透镜、反射镜等多种光学单元按一定次序组合成的系统，通常用来成像或者做一些简单的光学信息处理
- **空间采样**：图像传感器的像元的多少决定了空间的分辨率；像元越多，空间分辨率越高，图像越细腻、越清晰
- **图像传感器**由一个二维阵列组成，里面的每个元素称为像素，每个像素都能测量投射到他上面的光线数量并把它转化成电压信号，然后再通过后面的 AD 转换器转换成相应的数字信号
- 像素的总数目称为**空间分辨率**
- 分光棱镜将管线中的红、绿、蓝三个基本色分开，对每个颜色都单独投射到一个传感器元件上，将三个传感器原件收集到的三种单色信息合成可以产生效果非常好的彩色图像，但这样获得的彩色数字图像成本非常高，无法得到广泛的推广
- 改进后用一个传感器原件就可以产生彩色数字图像，将彩色滤光片像马赛克一样分布在传感器的所有像素上，是每个像素只能产生红、绿或蓝色三种颜色中的一种颜色的值，在经过相机处理单元处理以后依然可以输出一幅彩色的数字图像

1.3 计算机看到的不同类型的图像

- **黑白图像**：图像的每个像素只能是黑或者白，没有中间的过渡，故又称为**二值图像**
- **灰度图像**：指的是每个像素的信号值是由一个梁化后的灰度级来描述的图像，没有彩色信息，0 级表示黑色，255 级表示白色
- **彩色图像**：在 RGB 空间中，每幅彩色图像可以看成是由红 (R)、绿 (G)、蓝 (B) 三个不同颜色通道的灰度图像组合而成的
- 一张 RGB 空间的彩色图像可以分离出 R、G、B 三个通道分量，每个通道分离出的图像都是灰度图像，三个通道叠加后才产生彩色图像
- 在计算机眼里，每个通道实际上就是一个二维矩阵，矩阵中每个元素值就是一个像素值计算机眼里看到的实际上是一系列的二维数字
- HSI 颜色模型：H - 色调，S - 饱和度，I - 亮度
- HSI 与 RGB 之间的非线性映射：

$$I=\frac{1}{3}(R+G+B)$$

$$S=I-\frac{3}{R+G+B}[\min(R,G,B)]$$

$$H=\arccos\frac{(R-G)+(R-B)/2}{\sqrt{(R-G)^2+(R-B)(G-B)}}$$

2. 简单几何形状识别

2.1 角点检测

- **角点**在几何上通常可以被定义为**两条边的交点**，比如，三角形有三个角点，矩形有四个角点
- 在计算机视觉中，角点是**两个边缘的交点**，或者说是在局部区域内具有两个主方向的特征点

2.2 Harris 角点检测算法

- 假设在图中有一个滑动的小窗口，每次滑动都计算窗口内区域的平均灰度值
- 如果在各个方向上移动这个小窗口，窗口内区域的**平均灰度值都没有发生变化**，那么就认为在窗口内遇到了**背景**区域
- 如果在各个方向上移动这个小窗口，窗口内区域的**平均灰度值在某一特定方向（边缘方向）上没有发生改变**，那么就认为在窗口内遇到了**边缘**
- 如果在各个方向上移动这个小窗口，窗口内区域的**平均灰度值在各个移动的方向发生明显的改变**，那么就认为在窗口内遇到了**角点**
- 经过 Harris 角点检测算法，几何图形中的角点都被正确标记出来

2.3 识别多边形图形 - 箭头

- **第一种方法：角点特征的识别方法**
  - 采取的识别方式是基于角点特征的识别方法、基于模板匹配的识别方法、基于 HOG 特征提取的 SVM 分类方法。
  0. **预处理工作 - 提取标志牌，掩膜运算**
  i. **检测角点**
  ii. **焦点精确定位（聚类）**
  iii. **求出箭头的重心**
  iv. **求出箭头的顶端**
  箭头朝上和朝下情形： 顶点与重心的横坐标几乎相同，而纵坐标相差较远。利用这个特征，计算出每个角点与重心的横纵坐标差值，分别在横坐标和纵坐标用阈值进行排除即可筛选出顶点。
  箭头朝左和朝右情形：重心的纵坐标与顶点的纵坐标几乎相同，而横坐标相距较远，计算出重心与各个角点的横纵坐标的差值，用阈值分别在横坐标和纵坐标进排除，筛选出顶点。
  v. **完成判别**
  判断重心到顶点的向量方向，该方向即为箭头所指向的方向
  角点信息获得后，可以采用前面的决策树方法进行决策树设计，也可以用后面我们将学到的分类器方法，如：SVM 等进行处理。将提取到的特征向量与标准向量进行对比，从而完成对箭头方向的判别。
  vi. **分类判别**
  理想情况下，上、下、左、右四个方向箭头所对应的标准向量分别为：

$$e_1 = (0,1), e_2 = (0,-1), e_3 = (-1,0), e_4 = (1,0)$$

  以向上的箭头为例，令所提取到的特征向量为 α 。
  利用向量内积，求得 α 和四个标准向量的夹角，最终分类即为夹角最小的那一类。

$$\eta = \arccos \frac{\alpha \cdot e_i}{|\alpha||e_i|}$$

- **第二种方法：模板匹配的识别方法**
  - 模板匹配算法把模板图像在较大图像上进行滑动，并在每个位置比较模板和较大图像上相应区域的相似性，相似性最高的地方即所得的匹配结果。
  - 在模板匹配中，常用的相似性度量有以下几种：
    - **误差绝对值**：逐像素计算模板与原图的灰度差的绝对值，并求和
    - **平方差**：逐像素计算模板与原图的灰度的平方差，并求和。
    - **归一化平方差**：在平方差的结果基础上，除以原图和模板的灰度均值，以进行归一化。
    - **相关系数**：逐像素计算模板与原图的灰度的乘积，并求和。
    - **归一化相关系数**：在相关系数的结果基础上，除以原图和模板的灰度均值，以进行归一化。
  - 把模板图片从左到右滑动，从上到下滑动，并在每个位置计算模板图片和原图的误差绝对值的和
  - 模板匹配是图像处理中最基本、最常用的匹配方法。模板匹配具有自身的局限性，主要表现在它只能进行平行移动，若原图像中的匹配目标发生旋转或大小变化，该算法无效。
- **第三种方法：HoG 特征 + SVM 分类方法**
  - 对待识别图像提取其**方向梯度直方图** (Histogram of Oriented radient, HOG) 特征。
  - 将获得的特征图使用训练好的 SVM 分类器进行分类。

作业：

Harris.py:

```python
import cv2
import numpy as np
import os
from os import path

img_dir = "./fig" # 读取图像路径
savedir = "./results" # 保存图像路径

# 这三个参数即为本次探索参数，主要探索前两个，具体见ppt
for i in range(1,11):
        for j in  range(1,5):
                for k in range(1,5):
                        imagename="Harris_"+str(k)+".jpg"
                        image_name="Harris_"+str(k)+"_"+str(i).zfill(2)+"_"+str(2*j-1).zfill(2)+".jpg"
                        img = cv2.imread(path.join(img_dir,imagename))
                        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # 需要转换为灰度图
                        gray = np.float32(gray)
                        dst = cv2.cornerHarris(gray, i, j*2-1, 0.04)
                        img[dst>0.01 * dst.max()] = [0, 0, 255] # 检测到的角点画红色
                        os.makedirs(savedir, exist_ok="True")
                        cv2.imshow('corners', img)
                        print(path.join(savedir,image_name))
                        cv2.imwrite(path.join(savedir,image_name), img)
                        # if k==1:
                        #       sleep(3)
                        cv2.waitKey()
                        cv2.destroyAllWindows()
```