浙江大学

课程名称：计算机视觉

姓　　名：刘佳润

学　　院：计算机科学与技术学院

专　　业：数字媒体技术

学　　号：3180105640

指导教师：潘纲

2020 年 11 月 28 日

浙江大学实验报告

课程名称：计算机视觉 实验类型：综合

实验项目名称：制作个人小视频

学生姓名：刘佳润 专业：数字媒体技术 学号：3180105640

同组学生姓名： 指导老师：潘纲

实验地点： 实验日期：2020 年 11 月 28 日

 个人数码大头照……

一、 实验内容和要求

基于 OpenCV 生成小视频，制作有浙大元素的图片和个人信息的片头，自己设计情节，其中要缓慢地画一张画面，最后自己设计一个片尾。

做了一个火柴人初见了 OpenCV，尝试了一下，发现非常神奇，大受震撼，最后开心地跑走了的视频。

二、 实验器材

C++ OpenCV 4.5.0

开发平台：Visual Studio 2019 Debug x64

三、 具体实现

1. 基本图片与视频的读入

将一些图片和视频作为素材准备好，通过 C++的 fstream 库中定义的 _finddata_t 结构来获取文件结构，_findfirst 函数来检索指定目录下的文件，并通过文件的扩展名来获取文档，存入 vector 中。

对应的函数如下：

```
// 获取特定格式的文件名
void getAllFormatFiles(string path, vector<string>& files, string format)
{
    intptr_t hFile = 0;
    struct _finddata_t fileinfo;
    string p;
    if ((hFile = _findfirst(p.assign(path).append("\\*" + format).c_str(), &fileinfo)) != -1)
    {
        do
        {
            if ((fileinfo.attrib & _A_SUBDIR))
            {
                if (strcmp(fileinfo.name, ".") != 0 && strcmp(fileinfo.name, "..") != 0)
                {
                    //files.push_back(p.assign(path).append("\\").append(fileinfo.name) );
                    getAllFormatFiles(p.assign(path).append("\\").append(fileinfo.name), files, format);
                }
            }
            else
            {
                files.push_back(p.assign(path).append("\\").append(fileinfo.name));
            }
        } while (_findnext(hFile, &fileinfo) == 0);

        _findclose(hFile);
    }
}
```

图片可以通过 OpenCV 的接口读入并按帧写入视频中。写视频采用的是 cv::VideoWriter 类，可以通过流的形式将 Mat（帧）写入视频文件中。

```
img = imread(picFileList[1]);
resize(img, imgResized, Size(width, height));
for (int i = 0; i < fps; i++)
{
    writer << imgResized;
}
```

读取视频采用了 cv::VideoCapture 类，通过流的形式逐帧读入，进行 resize 以后再用 writer 逐帧写入。

```
capture.open(videoFileList[0]);
while (true)
{
    capture >> img;
    if (img.empty())
    {
        break;
    }
    resize(img, imgResized, Size(width, height));
    writer << imgResized;
}
capture.release();
```

2. 绘图：绘制 OpenCV logo 和火柴人

逐帧绘制的部分调用了 OpenCV 自带的一些几何体绘制函数，下面以 OpenCV 的 logo 绘制为例进行说明。

下面代码是一个红色部分的绘制。调用 ellipse 函数来画椭圆，参数说明如下：

void ellipse(InputOutputArray img, Point center, Size axes, double angle, double startAngle, double endAngle, const Scalar& color, int thickness = 1, int lineType = LINE_8, int shift = 0);

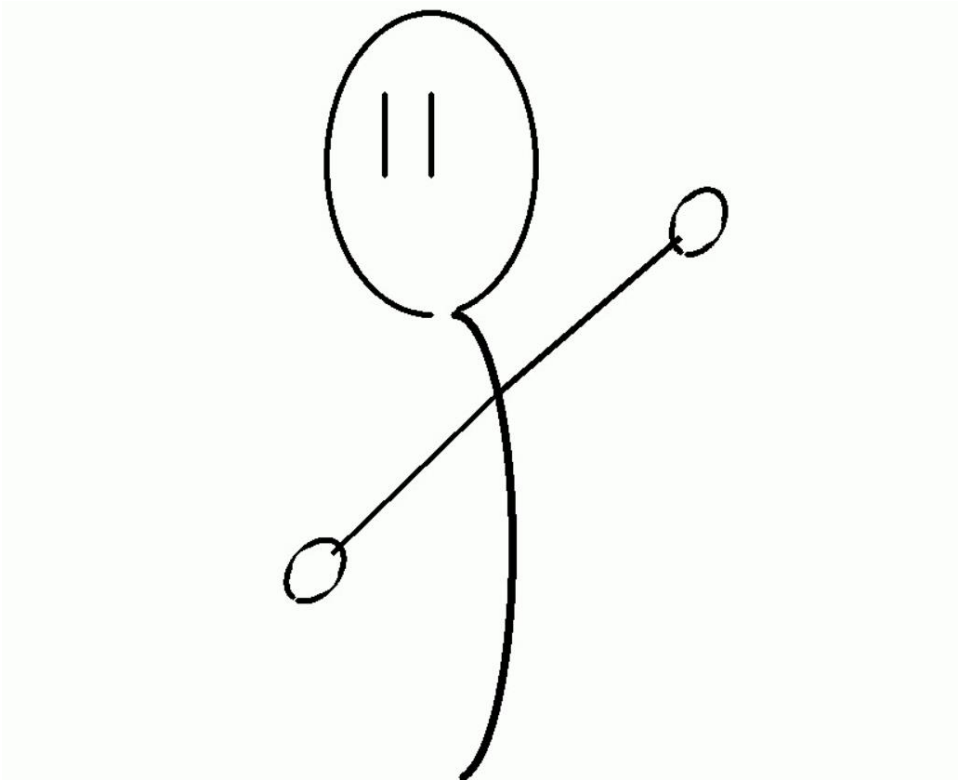
ellipse 函数将椭圆画到图像 img 上，椭圆中心为点 center，并且大小位于矩形 axes 内，椭圆旋转角度为 angle， 扩展的弧度从 0 度到 360 度，图形颜色为 Scalar(x， y，z)，线宽 (thickness)为 1，线型(lineType)为 8 (8 联通线型)。当线性设为-1 的时候为全填充。

为了达到逐渐绘制的效果，在扩展弧度的部分采用 fps 控制，在帧内逐帧绘制。

```
for (int i = 0; i < fps; i++)
{
    ellipse(img, Point(width / 2, height / 2 - 160), Size(130, 130), 125, 0, i * 290 / fps, CV_RGB(255, 0, 0), -1);
    circle(img, Point(width / 2, height / 2 - 160), 60, CV_RGB(0, 0, 0), -1);
    writer << img;
}
```

效果如下：

另一个简笔画是火柴人，效果如下：



3. 转场效果

编程实现了两种转场效果，一个是从模糊到清晰的过渡，一个是从右到左的进入。

模糊过渡调用 OpenCV 的 GaussianBlur 函数，根据 fps 控制逐帧从边缘到内部进行高斯模糊。

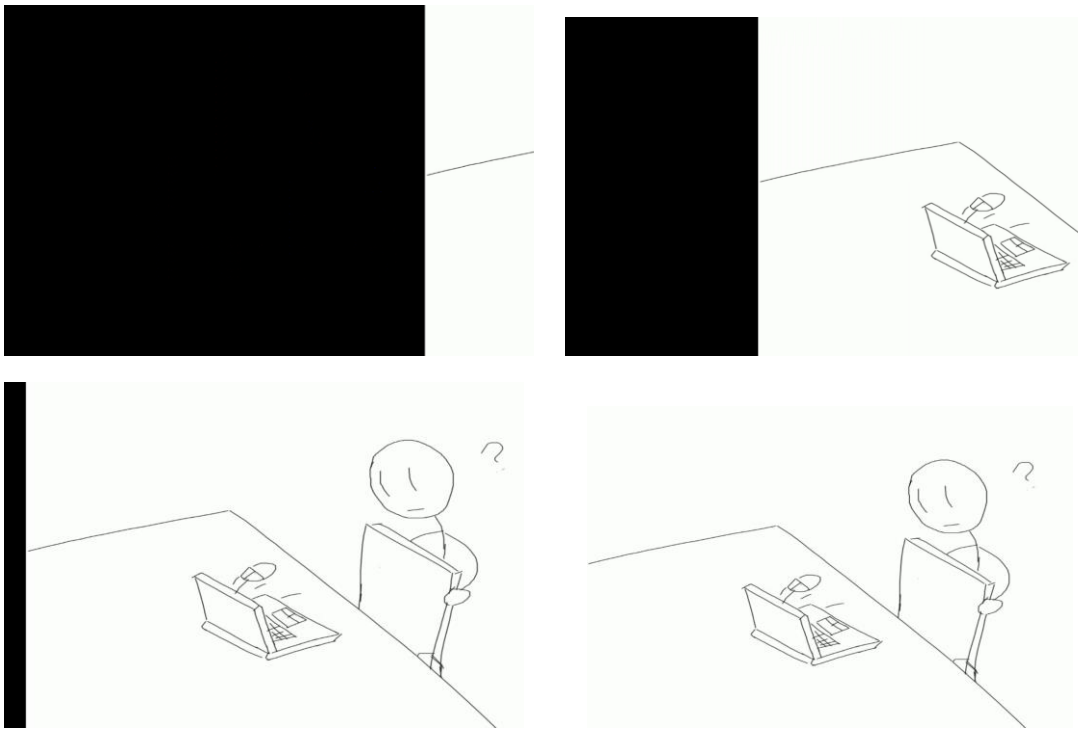
```
// 模糊转场效果
void blur(VideoWriter& writer, const Mat& img)
{
    Mat img_blur;
    for (int i = fps; i > 0; i--)
    {
        GaussianBlur(img, img_blur, Size(2 * i - 1, 2 * i - 1), 0);
        writer << img_blur;
    }
}
```

效果如下：

平移转场效果先设置一张空的图片，然后一列一列地将目标图片的像素复制到空图片的右侧，通过时间控制帧的写入。

```
// 右侧进入转场效果
void translate(VideoWriter& writer, const Mat& img)
{
    Mat newimg(img.size(), img.type());
    for (int i = 1; i <= fps; i++)
    {
        newimg.setTo(Scalar(0, 0, 0));
        Rect rect(img.cols * (1 - (float)i / (float)fps), 0, img.cols * (float)i / (float)fps, img.rows);
        img.colRange(0, img.cols * (float)i / (float)fps).copyTo(newimg(rect));
        writer << newimg;
    }
}
```

效果如下：



4. 播放与键盘控制

视频的播放采用 VideoCapture 类进行读入，逐帧用 imshow 函数进行输出。调用 OpenCV 的 waitKey 函数来判断键盘交互。

```
// 播放视频
void playVideo(string dir)
{
    VideoCapture capture;
    capture.open(dir);
    if (!capture.isOpened()) {
        cout << "Can't open video! " << endl;
        return;
    }
    int delay = fps;
    while (1) {
        Mat frame;
        capture >> frame;
        if (frame.empty())
            break;
        imshow("output", frame);
        if (delay >= 0 && waitKey(delay) >= 32)
            waitKey(0);
    }
    capture.release();
}
```

四、 实验结果与分析

实验结果请参见生成视频 output.mp4。其中一些重点的实现在上一部分也有

讲述。

但是在实验的过程中出现了一个现象：程序读取图片、视频和绘制的速度非常慢，整个一分多钟的小视频画完需要 45 秒左右。这一点后面还会再考虑是什么问题。（我认为可能是读取视频并解析的速度比较慢，因为不仅要逐帧读入、调整大小，还要逐帧写出）

五、 实验体会与心得

通过这次实验，又捡回了对于 C++的编程，感觉自己的代码变得规范点了（之前老写 python，写的有点乱）。之前我经常使用 Python 语言的 cv2 库，但是都是在做一些检测之类的 demo。这次实验熟悉了 OpenCV 的基本函数，掌握了对图像和视频的读写，还没有探索更多炫酷的 OpenCV 制作动画效果，因为 OpenCV 主要的用途还是在于辅助功能，而不是视频编辑功能。

在实验的过程中遇到了一个小问题就是“逐渐”绘制的过程，最后为了达到效果，查阅了一些资料，想到了用 for 循环结合 fps 来控制。最后达到了预期的效果。