浙江大学

计算机视觉(本科)作业报告

作业名称：制作个人视频

姓　　名：Orangeihero

学　　号：317010xxxx

电子邮箱：猜!

联系电话：猜!

导　　师：

2019 年 11 月 18 日

制作个人视频

一、 作业已实现的功能简述及运行简要说明

1.功能简述

编程生成视频，程序运行结束后会自动显示视频并自动保存在文件名为“video.avi”的视频文件，在运行程序观看视频的过程中，按空格可以暂停视频，再按一下空格可以继续观看视频。本视频除了一张个人照片外，没有从外部读入/载入任何图片或视频，实现了三个视频镜头切换。

视频的内容如下：

（1）片头：片头包含了我的个人照片及学号、姓名（英文）。

（2）第一个过渡：图片渐隐消失。

（3）第一幅简笔画：将视频一分为二，左边绘制太阳，右边绘制月亮。

（4）第二个过渡：太阳和月亮分别向左右移动显示出白色背景。

（5）第二幅简笔画：绘制一栋小房子。

（6）第三个过渡：小房子逐渐缩小至不见。

（7）片尾：个人图片、第一幅简笔画、第二幅简笔画和结尾画面分别变小各占据画面的四分之一，然后卷轴画面显示出结尾画面，最后结尾画面文字进行上下颤抖模糊，视频结束。

2.运行简要说明：

打开 Pycharm 运行程序，可以在运行程序过程中显示视频，并在程序文件夹下生成文件名为“video.avi”的视频文件。

二、 作业的开发与运行环境

Windows 10

Python 3.7.4

Opencv 4.1.1

三、 系统或算法的基本思路、原理、及流程或步骤等

程序共分为两大部分，即写视频和读取视频，接下来将分别介绍基本流程及原理。

1.写视频

（1）首先需要定义所要生成视频的参数，并制作贯穿始终的背景图片（我所选取的为纯白背景）。

（2）绘制片头（片头包含了我的个人照片及学号、姓名）：片头中用到的外部数据

为我的个人照片，绘制片头的步骤为读入图片并将其摆放在合适的位置，然后输入文本以显示出打字效果。

（2）绘制第一个过渡（图片渐隐消失）：改变图片的 RGB 值，使其分别接近 255，最后变为白色。

（3）绘制第一幅简笔画（太阳和月亮）：首先绘制一条垂直分割线，将视频区域一分为二；将左侧区域涂色为天蓝色，右侧区域涂色为深蓝色；然后在左侧绘制太阳及其光线并填色，在右侧绘制月亮并填色。

（4）绘制第二个过渡（太阳和月亮分别向左右移动显示出白色背景）：对矩阵进行操作改变矩阵的值，实现太阳和月亮向左右两边移动的效果。

（5）绘制第二幅简笔画（房子）：绘制房子的基本构造（屋顶、房身、窗户、门）并分别填色。

（6）绘制第三个过渡（小房子逐渐缩小至不见）：改变房子区域的大小并放置在背景图片的合适位置，变换图片以达到房子缩小的效果。

（7）绘制片尾（个人图片、第一幅简笔画、第二幅简笔画和结尾画面分别变小各占据画面的四分之一，卷轴画面显示出结尾画面，最后结尾画面文字进行上下颤抖）：使个人图片、第一幅简笔画、第二幅简笔画和结尾画面按顺序缩小分别占据图片的左上、右上、左下、右下区域，从左侧逐渐变幻出结尾画面（文字），最后结尾画面文字上下抖动模糊，最后趋于消失。

2. 读取视频

读取视频，即读取视频中的每一帧图片并连续显示，并通过设置按键实现视频的暂停和播放。

四、 具体如何实现，例如关键（伪）代码、主要用到函数与算法等

1. 写视频

（1）定义所要生成视频的参数，并制作贯穿始终的背景图片

①生成视频需要用到 cv2.VideoWriter(filename, fourcc, fps, frameSize[, isColor])函数，其中 filename 为保存的文件的路径，这里我直接保存在与程序相同的文件夹下，fourcc 为编码器，其本身是一个 32 位的无符号数值，用 4 个字母表示采用的编码器，这里采用“XVID”，fps 为要保存的视频的帧率，这里设为 25，frameSize 为要保存的视频的画面尺寸，这里设为（800，600），isColor 指示是黑白画面还是彩色画面。综上这里所运用的实例为

```
1.  out = cv2.VideoWriter(video_name, cv2.VideoWriter_fourcc(*'XVID'), fps,(800,
    600))
```

以初始化生成的视频。

②制作贯穿始终的背景图片，即一张 800*600 的纯白背景图片，将 RGB 三层数值全部设置为 255 即可，在这里运用实例为

```
new_image = np.zeros([600, 800, 3], np.uint8) + 255
```

（2）绘制片头（片头包含了我的个人照片及学号、姓名）：

①导入图片并改变图片大小以适应视频，导入图片运用了 cv2.imread(filename,flags)函数，filename 为导入图片的路径文件名，改变图像大小运用了 cv2.resize(InputArray src, OutputArray dst, Size, fx, fy, interpolation)函数，InputArray src 为输入图片的路径文件名，Size 为输出图片的尺寸，这里将图片缩放为 400*400 大小。具体应用如下：

```
image = cv2.imread('photo.jpg')
image = cv2.resize(image, (400, 400))
```

②在合适的位置改变矩阵的值，使个人照片显示在最中间，即在背景图片的宽 200~600，高 100~500 区域内放置图片。

③输入文本以显示打字效果

控制视频播放的速度，每三帧打出一个字显示在合适的位置，显示文本所用到的函数为 cv2.putText(img, text, org, fontFace, fontScale, color, thickness=None, lineType=None, bottomLeftOrigin=None)，其中 img 为要添加文字的图片，text 为添加的文字文本，org 为文字添加到图片上的位置，fontFace 为字体的类型，fontScale 为字体的大小，color 为字体的颜色，thickness 为字体的粗细。这里设置的实例为

```
cv2.putText(process, info[j], (text_x, 550), cv2.FONT_HERSHEY_SIMPLEX, 1, (0
, 0, 0), 1)
text_x += 20
```

每次显示一个字符，在每次显示完一个字符后，下一次添加文字的位置在原有的基础上向右挪动 20，相应的字符位置也加 1，这样即可以显示出打字的效果。

（3）绘制第一个过渡（图片渐隐消失）：

分为 20 帧来表示图片渐隐小时的过程，其主要算法为改变图片的 RGB 值，每次增加（255-该像素值）的 1/20，使其分别接近 255，到最后图片渐隐变为白色。

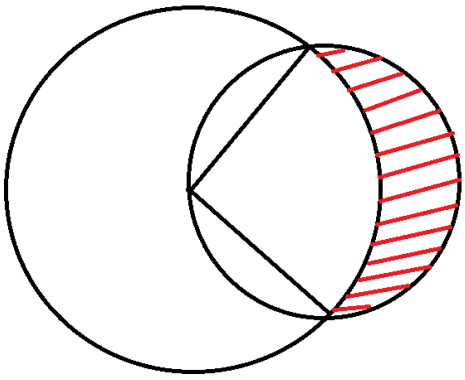
（4）绘制第一幅简笔画（太阳和月亮）：

首先调用 drawLine(out,img,startPoint,endPoint,color,thick,time, isVertical)函数进行垂直分割线的绘制：该函数是自定义函数，其中 out 表示要写的视频，img 表示需要画直线的图片，startPoint 是指线段的开始点，endPoint 表示线段的结束点，color 指线段的颜色，thick 指线段的厚度，time 指该动画播放的帧数，isVertical 表示绘制的线是否为垂直线。其具体算法为先调用 cv2.line(img,Point pt1,Point pt2,color,thickness=1,line_type=8 shift=0)函数，得到已经绘制直线的图片，然后根据 isVertical 判断该线为垂直还是水平，进行不同模式的覆盖。以垂直画线为例，在规定的帧数内对图片进行规定区域内垂直匀速更替，即将原图片对应部分替换为画完直线后图片的对应部分，显示出画直线的效果。

将左侧区域涂色为天蓝色，右侧区域涂色为深蓝色，其算法仍为采取用已涂色照片匀速覆盖未涂色图片以显示出涂色效果的覆盖原理，以下简称匀速覆盖算法。

在左侧绘制太阳及其光线，其中，调用 cv2.circle(img,center,radius,color, thickness=None, lineType=None, shift=None)函数画圆，其中 img 代表需要绘制圆的图片，center 代表圆心位置，radius 代表半径，color 代表圆的颜色，我们用此函数绘制以（200，300）为圆心，100 为半径的圆，然后采用匀速覆盖算法对其进行填色。

在右侧区域绘制月亮，，这里采用 cv2.ellipse(img, center, axes, angle, startAngle, endAngle, color, thickness, lineType, shift)函数，img 指需要添加椭圆的图片，center 指的是椭圆的中心，axes 指椭圆的长短轴的长度，angle 指椭圆的旋转角度，startAngle 指椭圆的起始角度，endAngle 指椭圆的结束角度，color 指绘制椭圆的颜色。此处调用该函数用两个圆弧拼起来一个月亮。如图所示，其中红色部分表示月亮的部分。



其中用到的函数实例如下

```
1.  process = cv2.ellipse(process, (400, 300), (200, 200), 0, 315, 405, (65, 214
    , 254), 2)
2.  process = cv2.ellipse(process, (541, 300), (141, 141), 0, 270, 450, (65, 214
    , 254), 2)
```

（5）绘制第二个过渡（太阳和月亮分别向左右移动显示出白色背景）：

对矩阵进行操作，匀速改变图片左右两半矩阵的值。我设置了 50 帧来显示以显示这个转场，对于左边的太阳，使左侧仍然涂色的部分，即横坐标 0:400-8*(j+1)的部分始终等于原图像 8*(j+1):400 的部分；对于右边的月亮，使右侧仍然涂色的部分，即横坐标 400+8*(j+1):800 的部分始终等于原图像 400:800-8*(j+1)的部分。把中间空出来的不需要涂色的部分用白色填充。

（6）绘制第二幅简笔画（房子）：绘制房子的基本构造（屋顶、房身、窗户、门）并分别填色。

调用上面所提到的绘制直线函数，绘制房子的结构，并用匀速覆盖算法给小房子填充颜色。

（7）绘制第三个过渡（小房子逐渐缩小至不见）：改变房子区域的大小并放置在背景图片的合适位置，变换图片以达到房子缩小的效果。

调用 cv2.resize（）函数，使小房子所在的区域大小匀速等比例变小，并将变小后的小房子赋值在白色背景的中央，在几次变小之后最后趋于消失在屏幕中央。

其中具体的代码实例如下：

```
image2 = cv2.resize(image2, (400 - 50 * i, 400 - 50 * i))
photo = new_image.copy()
photo[(100 + 25 * i):(500 - 25 * i), (200 + 25 * i):(600 - 25 * i), 0:3] = i
mage2
```

（8）绘制片尾

①使个人图片、第一幅简笔画、第二幅简笔画和结尾画面按顺序按比例缩小（使原尺寸为 800*600 的图像缩小为 400*300），分别占据图片的左上、右上、左下、右下区域。以结尾画面（结尾画面即为纯白背景图片上面打上两行结束语）为例，其实例代码如下：

```
out.write(end)
for i in range(25):
    change4 = change3.copy()
    image4 = cv2.resize(end, (775 - 16 * i, 575 - 12 * i))
    change4[25 + 12 * i:600, 25 + 16 * i:800, :] = image4
    out.write(change4)
```

其他代码的算法类似，只是赋值区域不同。

②从左侧逐渐变幻出结尾画面，即采用匀速覆盖算法，从左至右覆盖结尾画面，

③结尾画面文字上下抖动模糊，最后趋于消失。使用 cv.blur(src, ksize[, dst[, anchor[, borderType]]])函数，其中 src 表示需要进行操作的图像，ksize 表示模糊

核的大小，我通过改变 ksize 的大小来实现模糊的效果，最后趋于消失，代码实例如下：

```
change4 = cv2.blur(change4, (1, i + 1))
out.write(change4)
```

2. 读取视频

读取视频，即读取视频中的每一帧图片并连续显示，并通过设置按键实现视频的暂停和播放。使用 cv2.VideoCapture()函数读取视频，记录每一帧的图片并播放以形成视频的效果，通过设置按键实现视频的暂停和播放，具体操作代码实例如下：

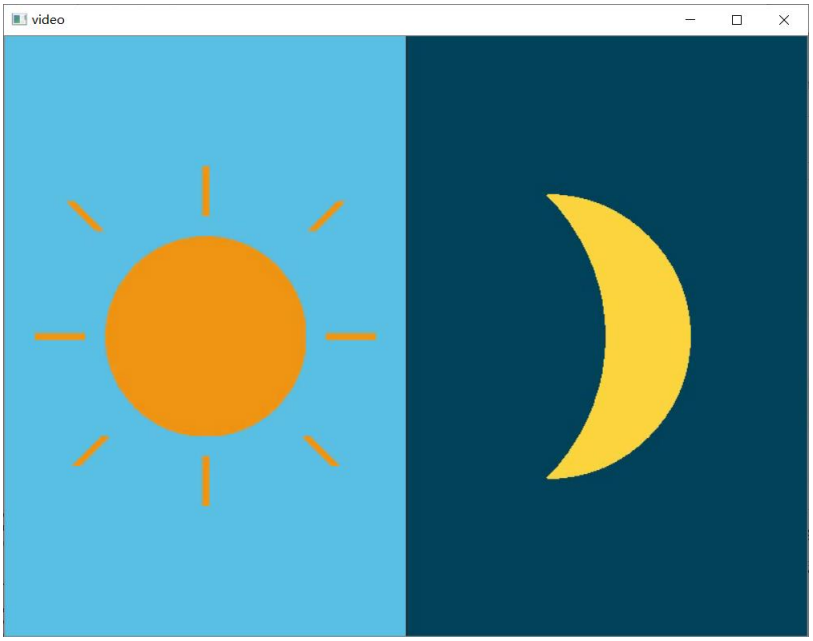
```
#按空格暂停和继续
if(k & 0xff == ord(' ')):
    cv2.waitKey(0)
```

五、 实验结果与分析

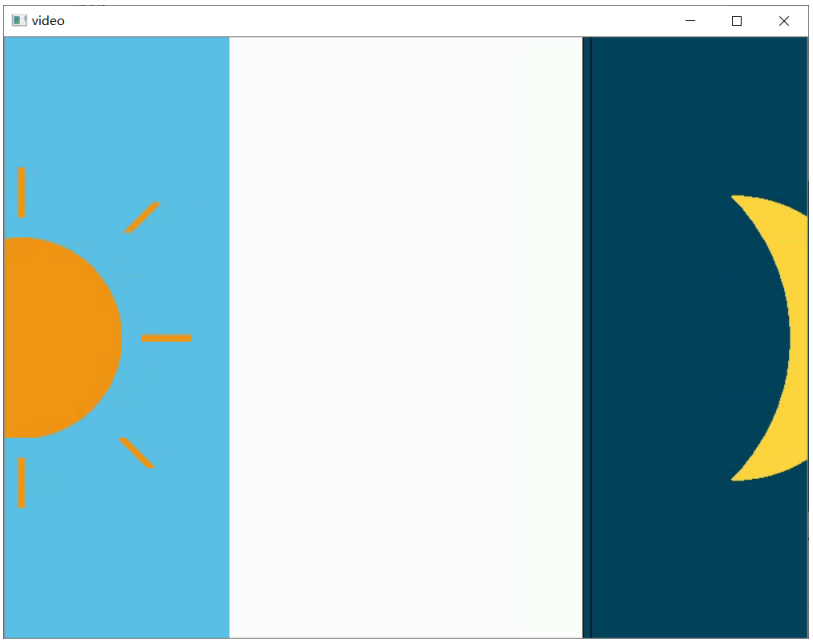
1. 片头

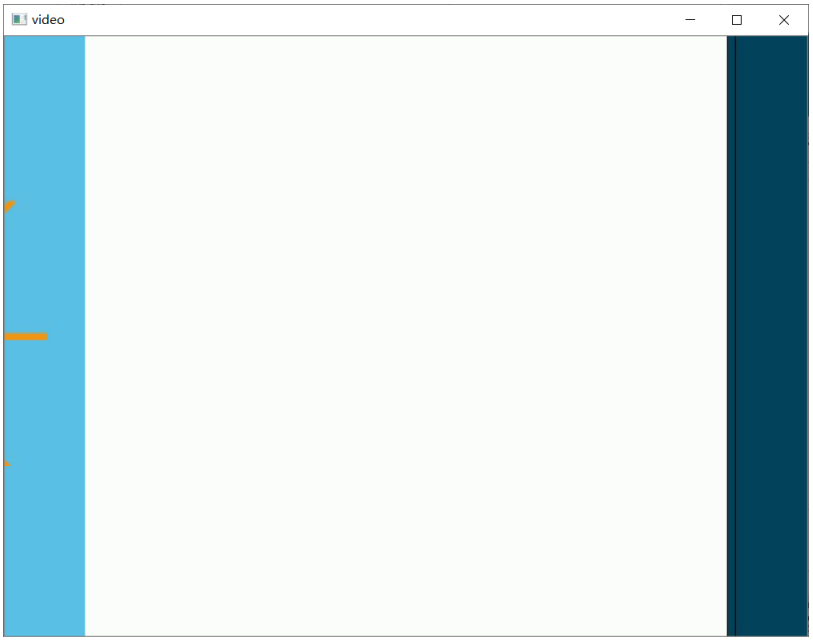
2. 第一个过渡：图片渐隐消失。

3. 第一幅简笔画：将视频一分为二，左边绘制太阳，右边绘制月亮

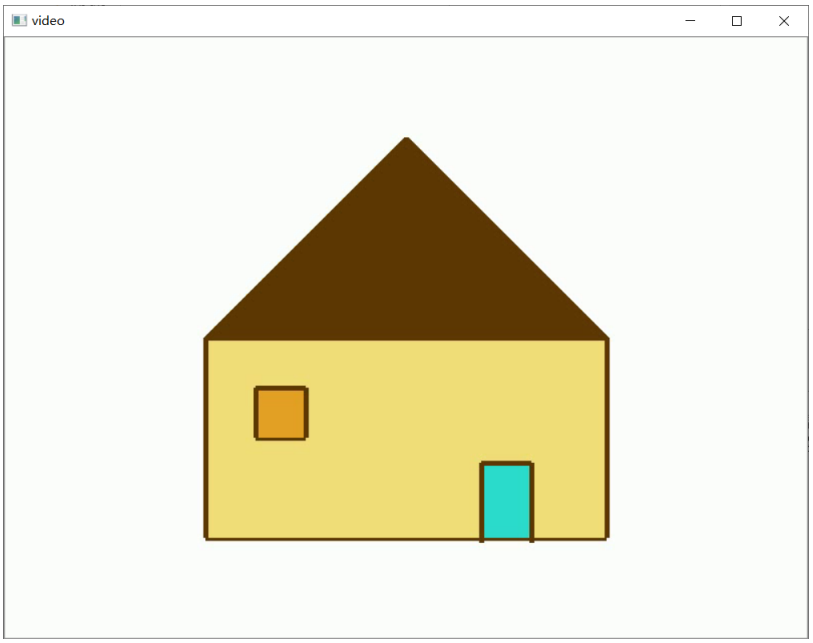


4. 第二个过渡：太阳和月亮分别向左右移动显示出白色背景。

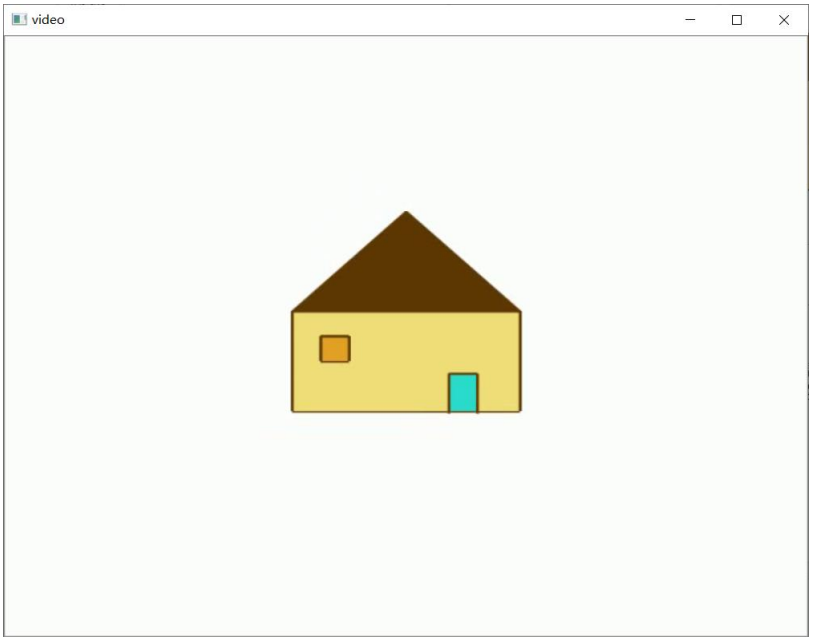


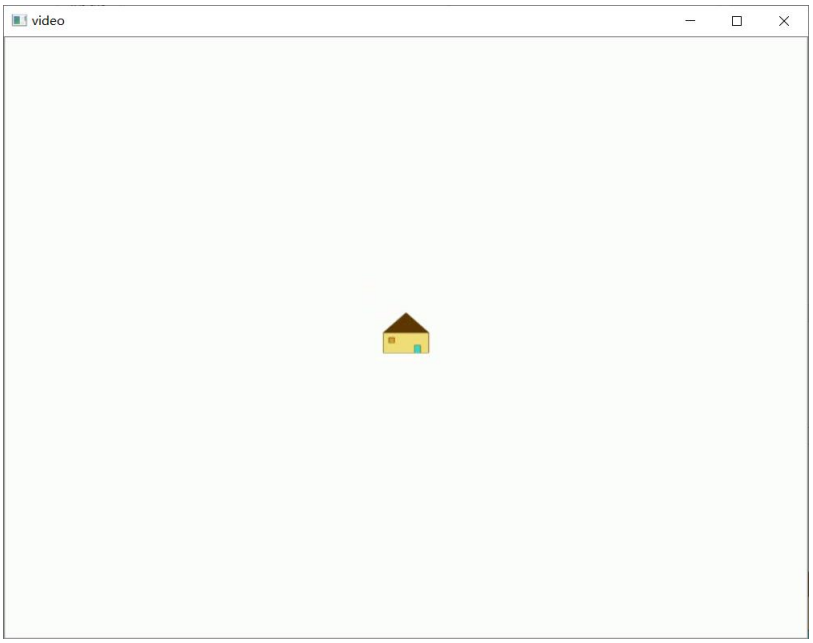


5. 第二幅简笔画：绘制一栋小房子。

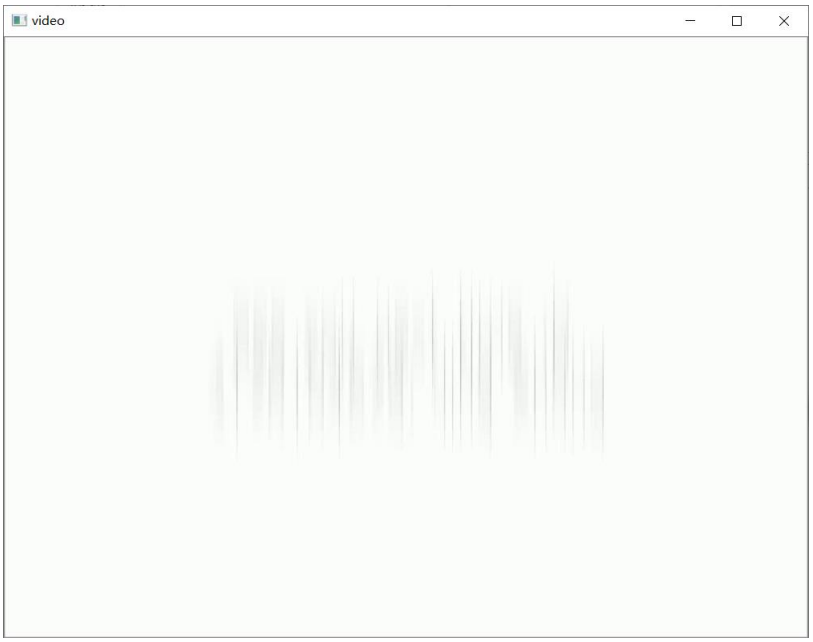


6. 第三个过渡：小房子逐渐缩小至不见





7. 片尾：个人图片、第一幅简笔画、第二幅简笔画和结尾画面分别变小各占据画面的四分之一，然后卷轴画面显示出结尾画面，最后结尾画面文字进行上下颤抖，视频结束。



具体实现效果请详见“video.avi”

六、 结论与心得体会

在完成本次实验的过程中我遇到了两个问题。第一个问题是在最开始的显示个人图片阶段，视频总是为0KB，之后发现问题存在于图片的尺寸与视频帧尺寸不匹配上，改变尺寸之后就可以写视频了。第二个问题出现在我的匀速覆盖算法上，在用已绘制颜色的图片缓慢代替现有图片时，我发现实际的视频会迅速显示出绘制完成的结果，之后会卡顿一段时间以显示我规定帧数的图片。这个问题困扰了我很长时间，后来我发现了问题所在：比如我们用original 来表示原有图片，此时我令 before = original，然后对 original 进行操作得到after 图片，然后将 before 缓慢过渡为 after，但是会出现上面的问题。后来发现，当对original 矩阵的一部分进行操作时，before 也会变化，所以会突然出现已涂色的照片，然后定格。解决方案是令 before = original.copy()，这样当都 original 矩阵的一部分进行操作时，before 不会发生变化，动画的过渡也会恢复正常。

另外一个小发现是，用函数读出图片后，三层通道顺序是 BGR，而不是 RGB。

通过本次实验，我了解了 Opencv 的基本操作及有关读取视频、写视频，读取图片、改变图片，绘制直线、圆、椭圆的基本操作，同时巩固了 Python 的语法及运用，通过不断的思考和改正完成了本次作业，感觉收获很大，受益匪浅。