浙江大学实验报告

专业：计算机科学与技术
姓名：吴同
学号：3170104848
日期：2019 年 11 月 21 日

课程名称：计算机视觉　指导老师：宋明黎　电子邮件：wutongcs@zju.edu.cn
实验名称：制作个人视频　实验类型：综合型　同组同学：无

1 Introduction

This report discusses an experiment to warm up for the use of the OpenCV library and master the basic skills of digital image processing with OpenCV by making a simple video.

In this experiment, a C++ program was implemented. The program receives a directory name as its parameter. It is asserted that all the image files in this directory are all in JPG format and there is a single video file in AVI format. The program reads these files and stitches them into a video in MP4 format. In the output video, a title displaying OpenCV logo, the input images and the input video are played one by one. There are transiting effects between images and my name and student ID are placed on the subtitle below.

2 Experimental Environment

- macOS Catalina 10.15
- clang version 11.0.0
- OpenCV version 4.0.1
- CMake version 3.13.4

3 Module Analysis and Result

3.1 Getting file names

To get file names under the specified dirent, UNIX system calls are used. In the **getFiles()** function, the input dirent name is checked first to ensure that the dirent exists and it really is a dirent. After the check passed, system call **opendir()** and **readdir()** are called to get the file names. Whenever a file name mathces the regex **.(jpg|JPG)$**, it will be pushed into the std::vector that stores the file names. Similarly, whenever a file name mathces the regex \\**.(avi|AVI)$**, it will be set as the input video name.

Figure **??** shows the file names displayed by the running program.

Figure 1: File names displayed

3.2 Making a title

The title of this video is a OpenCV logo. This logo is composed of three incomplete rings. To draw a ring, a sector is first drawed by calling **ellipse()**, and then a black cicle is drawed to dig the middle part. The effect of the animation is achieved by changing the angle. Figure **??** show the effects of **makeTitle()** function.

Figure 2: Test result of making title

3.3 Adding images

To add an image into the video, just call **cv::imread()** to read an image and write it into a video stream for several times. To add a subtitle, just call **cv::putText()**.

When an image is added to the video, **void transition(cv::VideoWriter& writer, const cv::Mat& img)** function is first called to generate a transition effect. This function is implemented as following:

```
typedef void transition_effect(cv::VideoWriter& writer, const cv::Mat&
    img);
void transition(cv::VideoWriter& writer, const cv::Mat& img)
{
   static transition_effect* transition_array[transition_num] = {
                                                 transition_effect0,
                                                 transition_effect1,
                                                 transition_effect2,
                                                 transition_effect3,
                                                 transition_effect4,
                                                 transition_effect5};
   static int i;
   (*(transition_array[i++]))(writer, img);
   if (transition_num == i)
   {
      i = 0;
   }
}
```

The **transition()** function uses six specific transition effects in turn. Each effect is implemented as a function which receives a cv::VideoWriter and a cv::Mat.

The first effect (figure **??**) is scaling. The image is gradually scaled up from the center of the canvas until it covers the entire canvas. In each frame, the image is scaled to the specified scale and then copied to a rectangle with the same size in the center of the canvas.

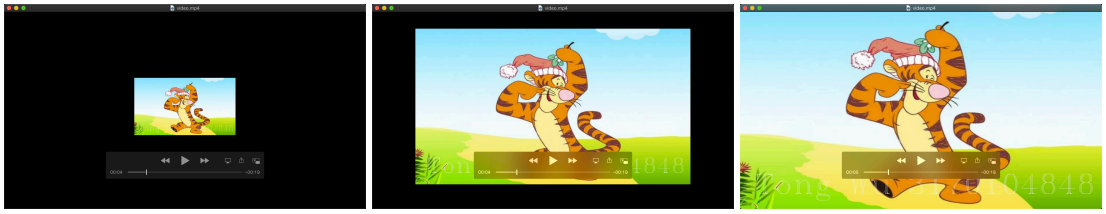

Figure 3: Transition effect 1

The second effect (figure **??**) is translation. The image slides from bottom to top into the canvas. In each frame, successive lines above the image are copied into a rectangular area of the same number of lines below the canvas.

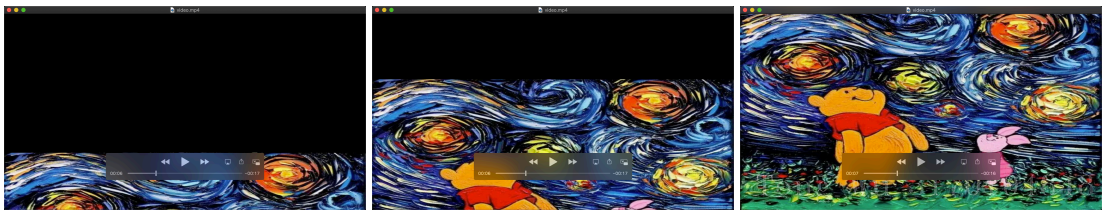

Figure 4: Transition effect 2

The third effect (figure **??**) is shutter. The image gradually emerges as if a shutter were open. In order to achieve this effect, there are black and white stripes in the mask. The width of the black and white stripes gradually changes, which leads the range of the image displayed to change.

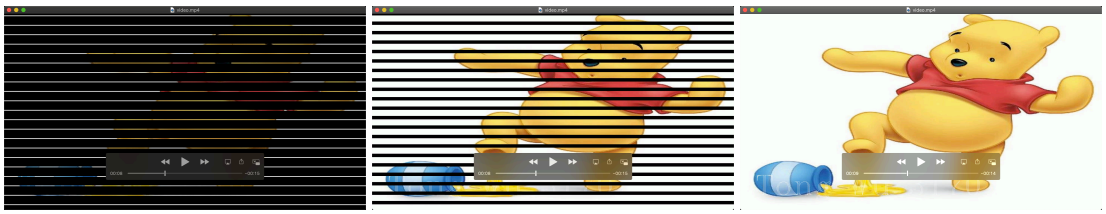

Figure 5: Transition effect 3

The fourth effect (figure **??**) is Gaussian blur. The image changes from blur to clarity. This function calls **GaussianBlur()** function in OpenCV and changes the degree of blur by changing ksize.

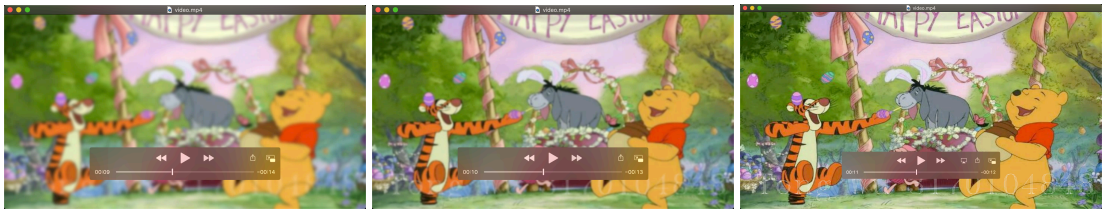

Figure 6: Transition effect 4

The fifth effect (figure 1) is erasing. A canvas with black and white gradient is pulled from right to left. There are two masks in the function, one for the original image and another for the canvas. The color of the stripes varies with space, and the width of the stripes varies with time.

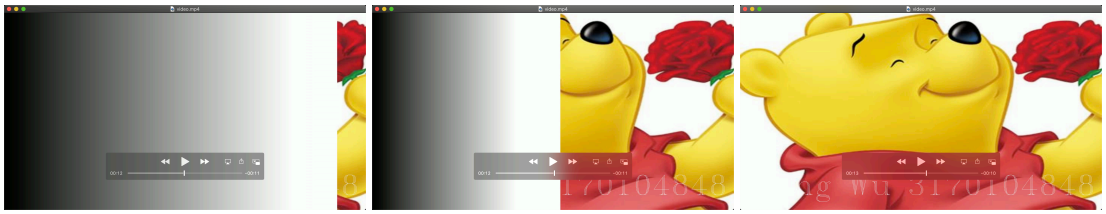

Figure 7: Transition effect 5

The sixth effect (figure 2) is binarization. The outline of the image is first drawed and then the image is colored by changing the threshold. Unlike general binarization, the binarization used here deals directly with colored images, showing the process of coloring.

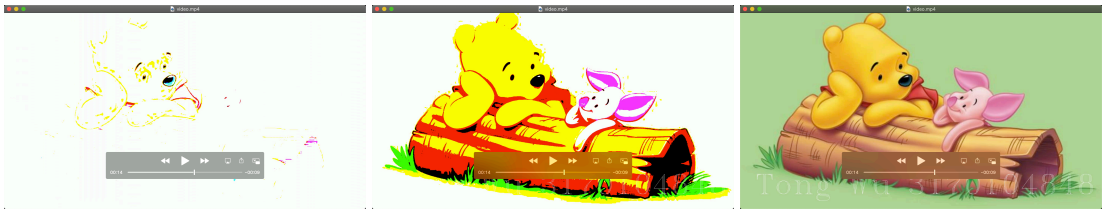

Figure 8: Transition effect 6

3.4 Adding a video

The stitching of the video is relatively simple. In the **writeAVI()** function, a cv::VideoCapture is constructed with the AVI file name. VideoCapture reads images from the input video stream frame by frame, and adds them to the output video stream after resizing them and adding subtitles.

In this experiment, I used the dragon logo as a test sample, as figure 3 shows.

Figure 9: Test result of video stitching

4 Review and Discussion

Through this experiment, I have become more proficient in the use of the OpenCV library and gained some skills in processing digital images. I have mastered the construction and use of several commonly used classes in OpenCV, such as Mat, rect, VideoCapture and VideoWriter. I have also understand the use of mask and some functions on cv::Mat by coding the transition functions.

There are also some shortcomings in my code. I used UNIX system calls to get the file names under a directory. In fact, there is a corresponding function in the OpenCV library, which can be used to make the code portable under different operating systems. In the processing of some images, algorithm can be improved to optimize running time.