Snehashis Pal
M.Tech CSE
Roll no: 2018201072
CSE578: Computer Vision
Assignment 0

All programs were written using C++ in linux with OpenCV installed on the system from the ubuntu repository. The functional parts of the code are provided in this pdf. The code is also available in the repository. GNU compiler collection(gcc) was use to compile all codes. The library requirements of opencv can be satisfied either by manually providing the libraries or through *pkg-config* utility as was done here.

I.a. Dividing image into frames

Every video is a set of temporally related images played one after another in a set period of time.
This task asks us to divide a video into its images using the opencv library. The general idea is read each frame of the video one by one and store them in a folder specified by the user.

The following procedure was used to achieve this.

1. Load the video using an instance of *VideoCapture*.
2. Read the frames of the video one by one by *VideoCapture::read*() storing each frame into a temporary 3 channel *Mat*.
3. Store the temporary *Mat* instance using *imwrite*().

All functions are a part of the opencv library. The program uses cmd line parameters as inputs.

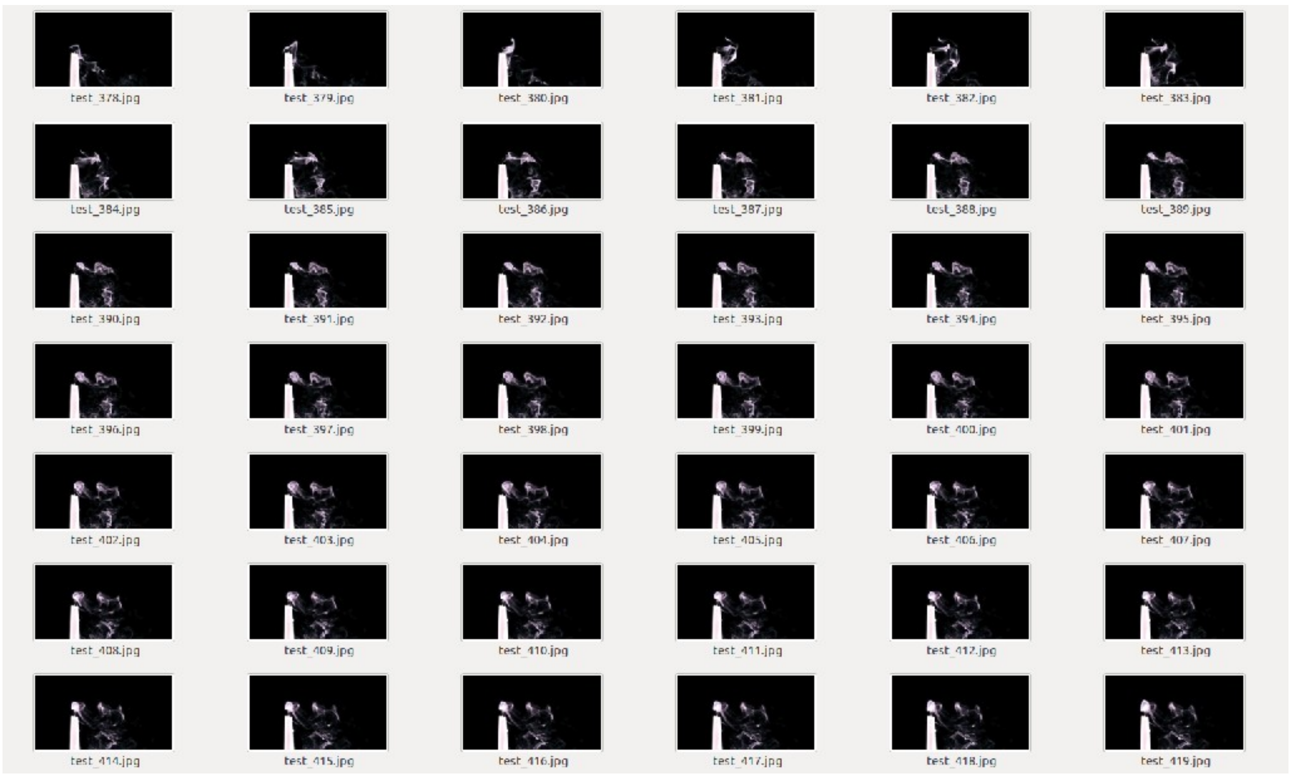
C++ Code

```
int main(int s,char *argv[]) {

    if(s != 3) printf("Format is splitvideo [SOURCE] [DEST FOLDER] \n");

    string filename = string(argv[1]);
    VideoCapture cap(filename);

    if(!cap.isOpened()) {
        printf("Error in opening file");
        return -1;
    }
    if(mkdir(argv[2],0777) == -1) {
        printf("Failed to create Directory");
        return -1;
    }

    Mat frame;
    string frameprefix = string(filename.c_str());
    frameprefix.replace(frameprefix.begin() + filename.rfind("."),frameprefix.end(),"");
    string dest = string(argv[2]) + "/" + frameprefix + "_";

    printf("Writing frames to %s as jpg images \n",argv[2]);

    while(cap.read(frame)) {
        int fno = cap.get(CV_CAP_PROP_POS_FRAMES)-1;
        string saveframe = dest + to_string(fno) + ".jpg";
        imwrite(saveframe.c_str(),frame);
        printf("Saved %s \n",saveframe.c_str());
    }

}
```

To run the program use the format : *splitvideo* [SOURCE VIDEO] [DESTINATION FOLDER].

Example.

Here is a section of the video https://www.youtube.com/watch?v=QwdHbh2kHUY as split frames.



I.b. Combining a set of images into a video.

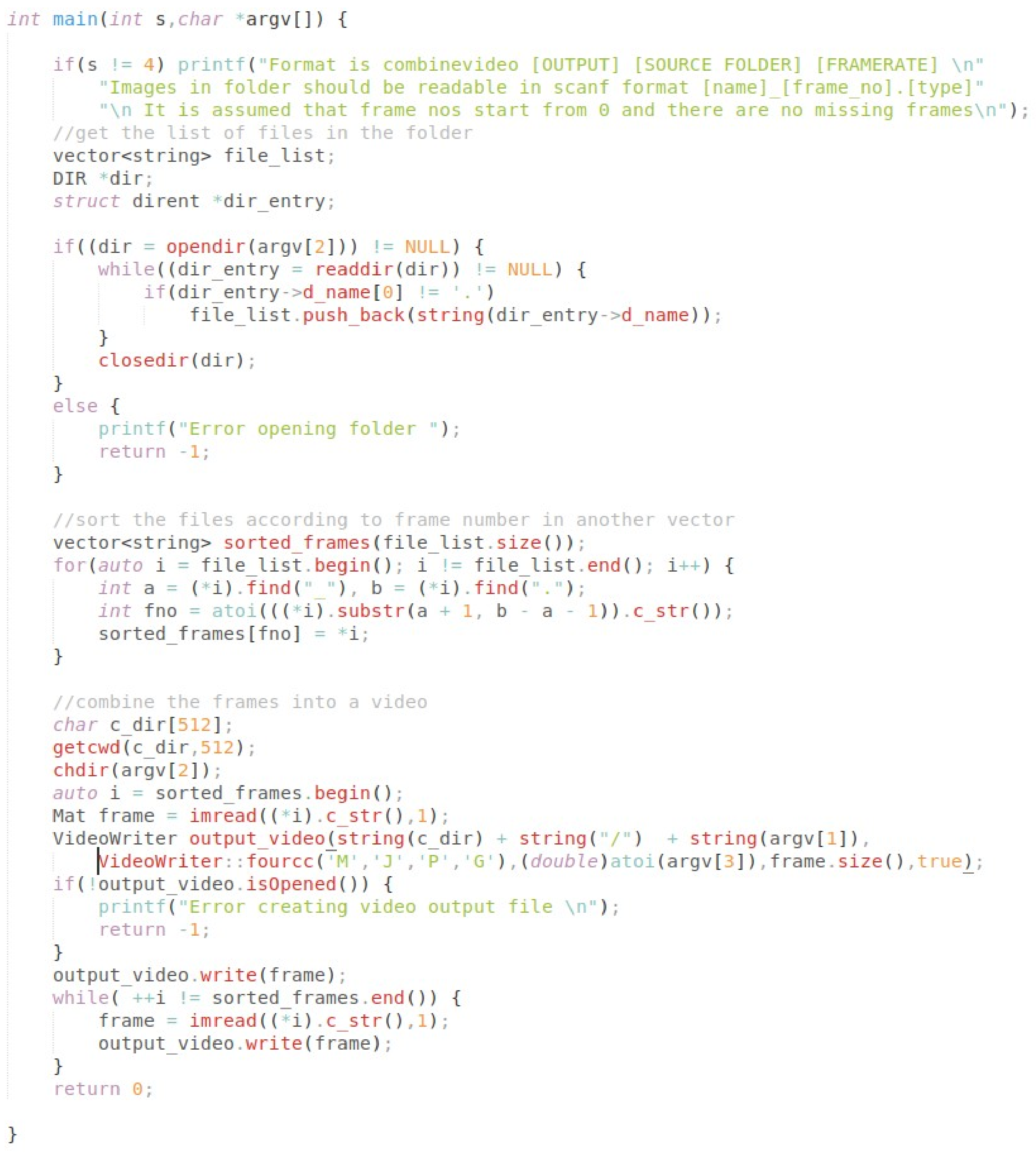
This is the opposite of a as we need to stich together a set of frames into a video. In the process we need to additionally provide the framerate of the resultant video to be created. OpenCV has the functionality of creating videos using an instance of the *VideoWriter* class from a set of images. However it is upto the programmer in which order the frames are sequenced. As a result we must have a standardized procedure to select which frame comes after one another.

- All the frames of a video are kept in a single folder.
- All the frames are named according to its position in the video i.e. if a frame image is the 6th frame in sequence in the video its name will be [name]_6.[ext].
- There are no skipped frame images i.e the set of frames numbers are contigous.

We need to read the frames as images one by one. However its is not guaranteed by the OS that the image read next from the folder will be the next frame in the sequence. Hence we need to explicitly find out the order of the images in sequence.

1. Read the image names from the folder one by one. The exact details are OS dependent. The code uses native system calls of the linux environment hence will only work in linux. Store them in a vector of image names.
2. Sort the image names according to their position as determined in their names. This can be done in linear time since all images are continous in the range 1 to n(no of frames) using direct addressing to store it in its correct place.
3. Once we have the sequence of images to read create a *VideoWriter* instance providing the compression coder to be used and the framerate as given by the user. Then use *VideoWriter :: write*() to store each frame into the video.

C++ Code



```cpp
int main(int s,char *argv[]) {

    if(s != 4) printf("Format is combinevideo [OUTPUT] [SOURCE FOLDER] [FRAMERATE] \n"
        "Images in folder should be readable in scanf format [name]_[frame_no].[type]"
        "\n It is assumed that frame nos start from 0 and there are no missing frames\n");
    //get the list of files in the folder
    vector<string> file_list;
    DIR *dir;
    struct dirent *dir_entry;

    if((dir = opendir(argv[2])) != NULL) {
        while((dir_entry = readdir(dir)) != NULL) {
            if(dir_entry->d_name[0] != '.')
                file_list.push_back(string(dir_entry->d_name));
        }
        closedir(dir);
    }
    else {
        printf("Error opening folder ");
        return -1;
    }

    //sort the files according to frame number in another vector
    vector<string> sorted_frames(file_list.size());
    for(auto i = file_list.begin(); i != file_list.end(); i++) {
        int a = (*i).find("_"), b = (*i).find(".");
        int fno = atoi(((*i).substr(a + 1, b - a - 1)).c_str());
        sorted_frames[fno] = *i;
    }

    //combine the frames into a video
    char c_dir[512];
    getcwd(c_dir,512);
    chdir(argv[2]);
    auto i = sorted_frames.begin();
    Mat frame = imread((*i).c_str(),1);
    VideoWriter output_video(string(c_dir) + string("/")  + string(argv[1]),
        VideoWriter::fourcc('M','J','P','G'),(double)atoi(argv[3]),frame.size(),true);
    if(!output_video.isOpened()) {
        printf("Error creating video output file \n");
        return -1;
    }
    output_video.write(frame);
    while( ++i != sorted_frames.end()) {
        frame = imread((*i).c_str(),1);
        output_video.write(frame);
    }
    return 0;

}
```

To run the program use the format : *combinevideo* [OUTPUT] [SOURCE FOLDER] [FRAMERATE]