Frame Transition Effect using Affine Transforms

Project Report

Tejas Nitin Joshi

ECEN 5653 Real Time Digital Media

1 Introduction

This project involves the usage of the ffmpeg software package which contains “libavcodec” which is the most updated audo/video codec library. FFmpeg in combination with the OpenCV library of programming functions for real time comuter vision, forms essential components of the project.

The algorithm uses the OpenCV library for performing “perspective” transforms which makes the transition effect apeear to be 3D like. The algorithm utilizes the usage of both bash script and the C++ code. The final output is a .mp4video with transformed frames but without losing any quality.

The Video used for this project is the “Sintel” ,which is an independently produced short film, initiated by the Blender Foundation as a means to further improve and validate the free/open source 3D creation suite Blender.

The lab requires a mpg2 video , but the ones available on the website was only .mkv format
The Matroska Multimedia Container is an open standard free container format, a file format that can hold an unlimited number of video, audio, picture, or subtitle tracks in one file.[1] It is intended to serve as a universal format for storing common multimedia content, like movies or TV shows. Matroska is similar in concept to other containers like AVI, MP4, or Advanced Systems Format (ASF), but is entirely open in specification, with implementations consisting mostly of open source software.

The original video can be downloaded from the below link

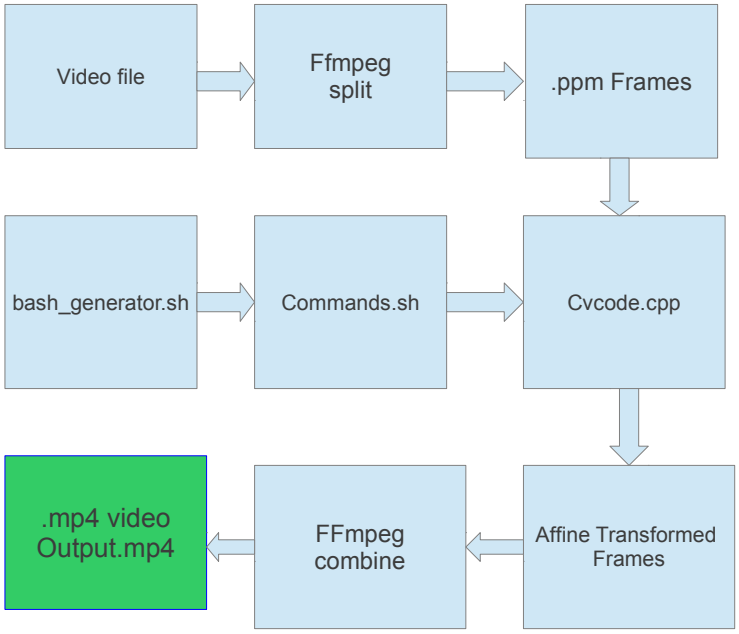
http://www.sintel.org/download

I am using the HD.720p quality file available on the above link for this project.

2 Program Flow:

Below Diagram describes the flow of the program

File conversion and generation flow



The ffmpg software is used to convert the mkv video to Portable Pixel Map(.ppm) image files. The bash generator generates commands.sh file which then needs to be converted to a bash file. This bash file calls cvcode.cpp by passing arguments for different “warp degrees” and “image file names”. This causes the cvcode.cpp to grab a frame and convert it accordingly to the warp degree provided to it. The frame progression for warping is as explained in the fowllowing section.

The actual implementation is a bit different it involves four parts,

- The generation of .ppm frames from the video file.
- The generation of the cvcode executable
- Runng the bash script that calls the cvcode executable whic converts frames.
- Running the ffmpeg command to combine the transformed frames to a mp4.video

3 Steps to Re-Create the Transformation:

1. Create a folder , lets say “ImgTrans” copy the video file(Sintel.720p.mkv), the bash_generator.sh, and the cvcode.cpp and CMakeLists.txt files.

2. Create a folder “og_frames” inside the “ImgTrans” folder.

```
drix@lynx:~/prog$ mkdir ImgTrans
drix@lynx:~/prog$ cd ImgTrans/
drix@lynx:~/prog/ImgTrans$ mkdir og_frames final_frames
drix@lynx:~/prog/ImgTrans$
```

   (The bash scrip does this for you if you dont)

3. Next step is to generate the frames using the ffmpeg command(Your present working directory must always be ImgTrans for this to work through the project.)

```
ffmpeg -t 125 -i "Sintel.2010.720p.mkv" "./og_frames/output_%05d.ppm"
```

   This will generate all the frames in the og_files folder for 2 min of the video.

   -an arguments strips away only video and discards audo track

   -t is the time in seconds for which the conversion starts from the beggning of the file

   -i flag that indicates input file is being specified next

4. Next you must perform the following commands to generate the cvcode executable from CMakeLists.txt and cvcode.cpp while the frames are being processed by opening a second terminal window

   cmake .

   make

5. Once the cvcode executable is ready, The final Step is to execute the bash_generator.sh script (let the frame conversion finish first)

   This script accepts 2 arguments as below

```
./bash_generator.sh <number_of_frames> <horizontal_pixel_resolutio>
```

   the first argument is the number of frames created in the og_frames folder and the second argument is the horzontal pixel resolution for the video file. (in this case 1080p)

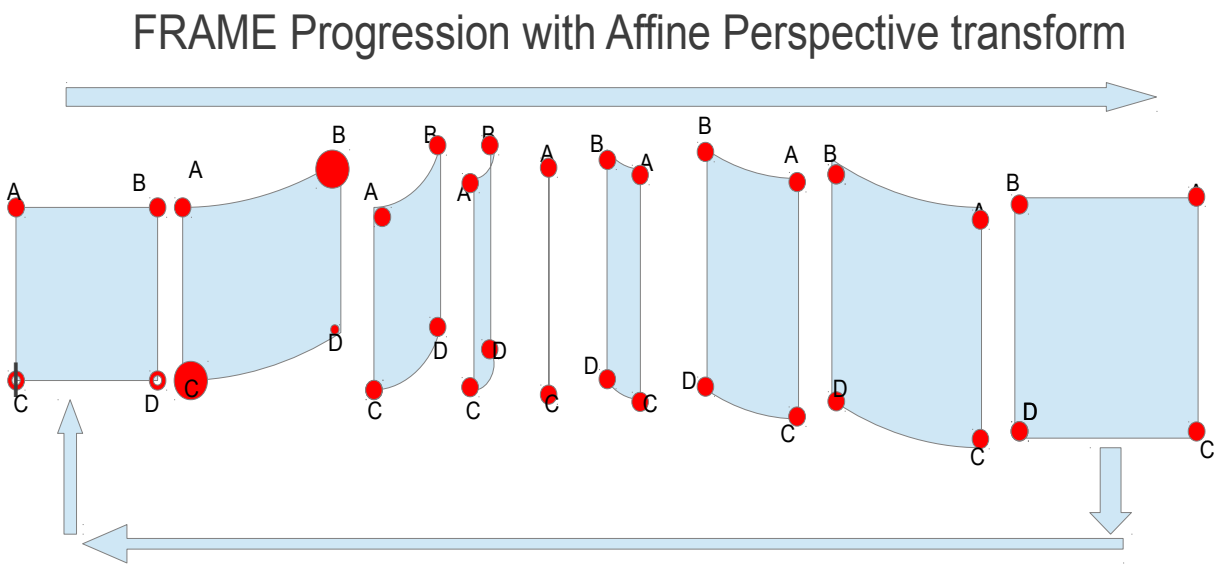
6. This will generate the transformed frames by calling the cvcode executable in the “final_frames” folder.

7. Finally the ffmpeg combine command will generate the output .mp4 file

```
ffmpeg -i ./final_frames/output_%05d.ppm -vcodec mpeg4 -sameq cool.mp4
```

   In this case my file output file name is cool.mp4 -vcodec flag specifies mpeg4 codec to be used.

4 What the algorith does to the Video:



Above diagram shows the frame transformation progression in the final video. The frame is basically being perspective changed 180 degrees and then flipped back to be continued in the same progression again.

This is possible with the use of the OpenCv command for perspective and Affine Transformation.

getPerspectiveTransform

Calculates a perspective transform from four pairs of the corresponding points.

C++: Mat **getPerspectiveTransform**(InputArray **src**, InputArray **dst**)

C++: Mat **getPerspectiveTransform**(const Point2f **src**[], const Point2f **dst**[])

Python: cv2.**getPerspectiveTransform**(src, dst) → retval

C: CvMat* **cvGetPerspectiveTransform**(const CvPoint2D32f* **src**, const CvPoint2D32f* **dst**, CvMat* **map_matrix**)

Python: cv.**GetPerspectiveTransform**(src, dst, mapMatrix) → None

Parameters:
- **src** – Coordinates of quadrangle vertices in the source image.
- **dst** – Coordinates of the corresponding quadrangle vertices in the destination image.

The function calculates the 3×3 matrix of a perspective transform so that:

$$\begin{bmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{bmatrix} = \texttt{map\_matrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

where

$$\texttt{dst}(i) = (x'_i, y'_i), \texttt{src}(i) = (x_i, y_i), i = 0, 1, 2, 3$$

See also: `findHomography()`, `warpPerspective()`, `perspectiveTransform()`

The tough part is calculating of the movement of the 4 coordinates of the frame (i.e.A, B C, D) to get the resultant frame position.

Once the Affine transformatin matrix is obtianed the OpenCV function warpPerspective() is used which accepts the source image matrix, its size and the transformation matrix and stores it in the specified destination matrix

Imread and Imwrite functions are used to read and write images to and from the .ppm files.

5 WHAT the Bash Script Does?

The bash_generator.sh accepts the frame count and horizontal resolution(of video file) as the arguments in that order.

Thiis creates a new file called commands.sh and generates command

```
"./cvcode ./og_frames/output_%05d.ppm %d 0"
```

where %05d is replaced with the 5 digit sequence number of the file name.

Note: I tried attempting to do this via C code but inserting a inteiger by converting it to a string with padded zeros would give me errors, i tried different libraries and other tricks to do this but that did not work out for me. Hence i had to resort to bash script.

Once the commands.sh file is created it pre-appedds the hash-bang identifier at the top of the file and gives it executabel permission.

Then it executes the commands.sh file which calls on the “cvcode” executable to peform affine transformation operations on each generated frame.

The output files are available in the “final_frames ” folder.

6 What the cvcode.cpp contains? And what it does?

The cvcode.cpp is consists of the routine which accepts the following arguments

```
./cvcode   <filename.ppm> <warp_degree> <rotation_degree>
```

The arguments are provided by the commands.sh bash script which calls the executable.

This code uses the standard i/o libraries along wit the openCV libraries and the system libraries for directory navigation and creation (unistd.h),

The code reads the file and stores it into a Matrix“ Mat src” obtains its dimensions from the src.rows and src.cols structure properties for the image.

The code then parses the arguments. The <rotation_degree > argument has not been implemented, but that is a option for future combinational transition effects.

And defines expressions for the coordinate points of the source frame and then goes on to define the destination coordinates.

The getPerspectiveTransform() function returns the Afiine matrix required to warp the image

and the warpPerspective() function uses the above generated matrix to perform Warping on the frame.

The rest of the code changes directory to write the processed frame to the “final frames ” folder and then returns.

7 Timing, Bitrate and FPS Analysis:

I did not implement any acceleration as most of my time was consumed when i was attempting to write code to pass “file_name”strings to the cvcode function. I lost time there. Hence i performed some timing Analysis and Bitrate and FPS.

I measured the frame conversion time for the cv code to take all frames and perform Affine transforms was about **17min and 32 seconds(approx).**

The conversion took up most of the time.

The frame splittting and combinining them back had lesser time compared to the conversion time.

Individual Frame was **2.1Mb** .ppm file and hence would require a lot of time to scan through to perform the transform.

Source File Information:

```
drix@lynx:~/prog/extende/code$ avprobe Sintel.2010.720p.mkv
avprobe version 0.8.6-4:0.8.6-0ubuntu0.12.04.1, Copyright (c) 2007-2013 the Liba
v developers
  built on Apr  2 2013 17:02:36 with gcc 4.6.3
[matroska,webm @ 0x18a67a0] max_analyze_duration reached
[matroska,webm @ 0x18a67a0] Estimating duration from bitrate, this may be inaccu
rate
Input #0, matroska,webm, from 'Sintel.2010.720p.mkv':
  Duration: 00:14:48.03, start: 0.000000, bitrate: 640 kb/s
    Chapter #0.0: start 0.000000, end 103.125000
    Metadata:
      title           : Chapter 01
    Chapter #0.1: start 103.125000, end 148.667000
    Metadata:
      title           : Chapter 02
    Chapter #0.2: start 148.667000, end 349.792000
    Metadata:
      title           : Chapter 03
    Chapter #0.3: start 349.792000, end 437.208000
    Metadata:
      title           : Chapter 04
    Chapter #0.4: start 437.208000, end 472.075000
    Metadata:
      title           : Chapter 05
    Chapter #0.5: start 472.075000, end 678.833000
    Metadata:
      title           : Chapter 06
    Chapter #0.6: start 678.833000, end 744.083000
    Metadata:
      title           : Chapter 07
    Chapter #0.7: start 744.083000, end 888.032000
    Metadata:
      title           : Chapter 08
    Stream #0.0(eng): Video: h264 (High), yuv420p, 1280x544, PAR 1:1 DAR 40:17,
24 fps, 24 tbr, 1k tbn, 48 tbc
    Stream #0.1(eng): Audio: ac3, 48000 Hz, 5.1, s16, 640 kb/s
```

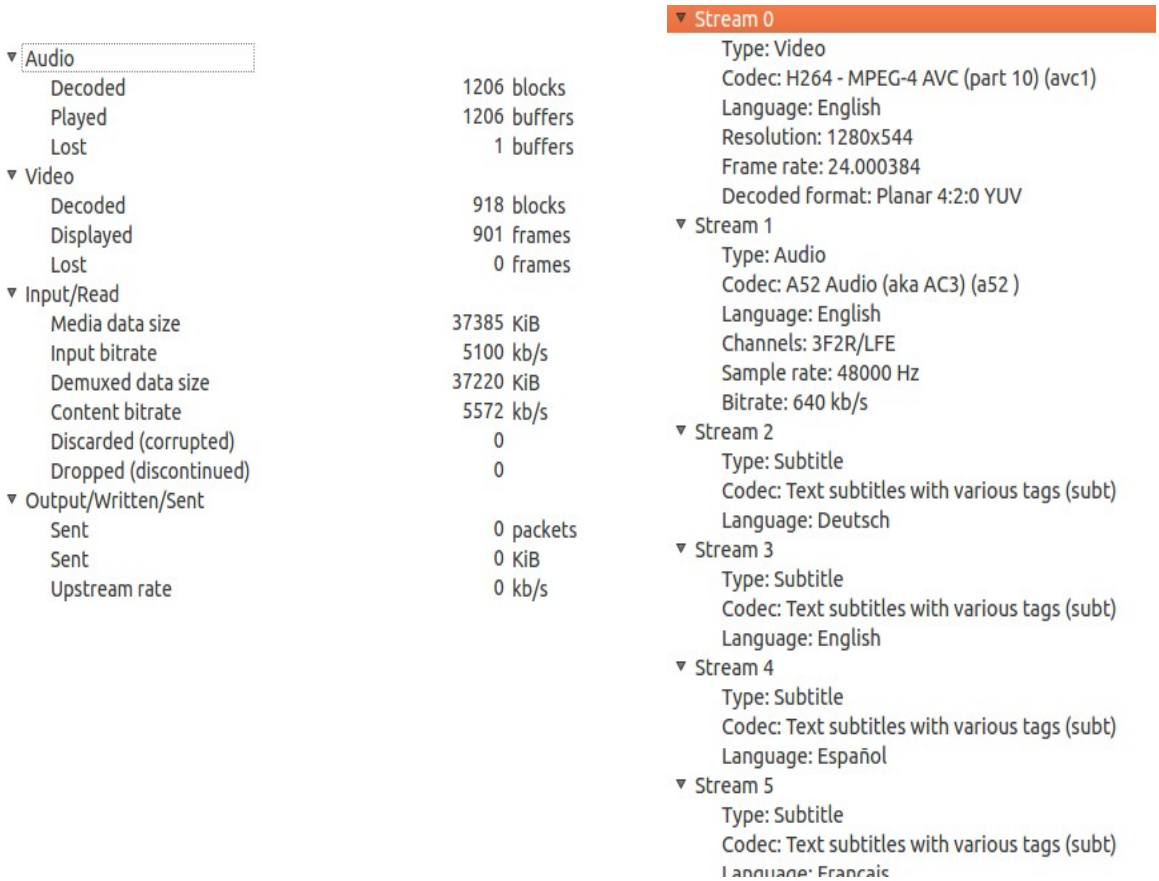
Metadata:

```
    Metadata:
      title           : AC3 5.1 @ 640 Kbps
    Stream #0.2(ger): Subtitle: [0][0][0][0] / 0x0000
    Stream #0.3(eng): Subtitle: [0][0][0][0] / 0x0000
    Stream #0.4(spa): Subtitle: [0][0][0][0] / 0x0000
    Stream #0.5(fre): Subtitle: [0][0][0][0] / 0x0000
    Stream #0.6(ita): Subtitle: [0][0][0][0] / 0x0000
    Stream #0.7(dut): Subtitle: [0][0][0][0] / 0x0000
    Stream #0.8(pol): Subtitle: [0][0][0][0] / 0x0000
    Stream #0.9(por): Subtitle: [0][0][0][0] / 0x0000
    Stream #0.10(rus): Subtitle: [0][0][0][0] / 0x0000
    Stream #0.11(vie): Subtitle: [0][0][0][0] / 0x0000
Unsupported codec with id 94210 for input stream 2
Unsupported codec with id 94210 for input stream 3
Unsupported codec with id 94210 for input stream 4
Unsupported codec with id 94210 for input stream 5
Unsupported codec with id 94210 for input stream 6
Unsupported codec with id 94210 for input stream 7
Unsupported codec with id 94210 for input stream 8
Unsupported codec with id 94210 for input stream 9
Unsupported codec with id 94210 for input stream 10
Unsupported codec with id 94210 for input stream 11
drix@lynx:~/prog/extende/code$
```

The video has a resolution of **1280p x544** and is a **H264 (High)** quality video with a average **24 FPS**

It has multiple audio streams for different languages as seen in the Metadata.

Stattistics from VLC and Codec Information:



▾ Audio	
Decoded	1206 blocks
Played	1206 buffers
Lost	1 buffers
▾ Video	
Decoded	918 blocks
Displayed	901 frames
Lost	0 frames
▾ Input/Read	
Media data size	37385 KiB
Input bitrate	5100 kb/s
Demuxed data size	37220 KiB
Content bitrate	5572 kb/s
Discarded (corrupted)	0
Dropped (discontinued)	0
▾ Output/Written/Sent	
Sent	0 packets
Sent	0 KiB
Upstream rate	0 kb/s

Output Video mp4 Information:

```
drix@lynx:~/prog/extende/ImgTrans$ avprobe cool.mp4
avprobe version 0.8.6-4:0.8.6-0ubuntu0.12.04.1, Copyright (c) 2007-2013 the Liba
v developers
  built on Apr  2 2013 17:02:36 with gcc 4.6.3
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'cool.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2mp41
    encoder         : Lavf53.21.1
  Duration: 00:01:57.24, start: 0.000000, bitrate: 2944 kb/s
    Stream #0.0(und): Video: mpeg4 (Simple Profile), yuv420p, 1280x544 [PAR 1:1
DAR 40:17], 2943 kb/s, 25 fps, 25 tbr, 25 tbn, 25 tbc
```

The resolution is unchanged and the average frame rate is **25FPS** with corresponding increase in bitrate. There are no other streams as only the video was extracted.

Video Information from VLC and Codec information:

▼ Stream 0
- Type: Video
- Codec: MPEG-4 Video (mp4v)
- Resolution: 1280x544
- Frame rate: 25
- Decoded format: Planar 4:2:0 YUV

▼ Audio		
Decoded	0	blocks
Played	0	buffers
Lost	0	buffers
▼ Video		
Decoded	1117	blocks
Displayed	1103	frames
Lost	0	frames
▼ Input/Read		
Media data size	12889	KiB
Input bitrate	4688	kb/s
Demuxed data size	12859	KiB
Content bitrate	4594	kb/s
Discarded (corrupted)	0	
Dropped (discontinued)	0	
▼ Output/Written/Sent		
Sent	0	packets
Sent	0	KiB
Upstream rate	0	kb/s

8 Conclusion:

Learing about the ffmpeg software and the video and audio transport streams was very interesting. Initally i had atempted to do a complete C++ code for the execution but got tangled in a issue with generation of file names. Specifically, trying to insert a intiger (converted to a string) in between a string.

I tried attempting to write code for that using pointers, using the inbuilt string functions(strcpy(), strcat() and such). ended up hunting on the web for this and saw example of bash script and went ahead and coded for it.

My final project size was appproximately greater than 12GBs and had to remove the contents of the "og_folder" which contained the .ppm frames extracted from the original video (converted frames can be found in "final_frames" folder). to bring it down to 6.7Gbs which i have added to the SD card.

Also i apologize if i did not stick to the input file format requirement of mpeg2 for the lab but Sintel was too visually stunning to let go :) .

Proposed Improvement's :

- I will further work on this to turn it into one streamlined package and figure out the issue with the string generation i faced, eliminating the bash scripts.
- Implementation of image scaling, rotationg and translation transforms can be added for more visual 3D effects.
- Would also llie to recode it and create a parallel version and see how it fast would work on a CUDA,GPU.

9 References:

1. Converting video to image sequence and back:

   http://pr0gr4mm3r.com/linux/convert-video-to-images-and-back-using-ffmpeg/

2. Lossless reconversion to the video format :

   http://stackoverflow.com/questions/4839303/convert-image-sequence-to-lossless-movie

3. Extracting audio:

   http://linuxpoison.blogspot.com/2010/04/how-to-extract-audio-from-video-file.html

   http://zoid.cc/ffmpeg-audio-video/

4. OpenCV functions for perspective Transformations and example for warp affine

   http://docs.opencv.org/modules/imgproc/doc/geometric_transformations.html

   http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/warp_affine/warp_affine.html

5. Re-directing bash script output

   http://www.theunixtips.com/bash-self-redirect-of-scripts-output-to-file

6. ffmpeg time based extration of frames

   http://software.intel.com/en-us/articles/using-intel-vtune-performance-analyzer-and-intel-integrated-performance-primitives-for-real-time-media-optimization/

7. Other usefull links for bash scripts

   http://superuser.com/questions/246837/how-do-i-add-text-to-the-beginning-of-a-file-in-bash

   http://stackoverflow.com/questions/1117134/padding-zeros-in-a-string

   http://www.cyberciti.biz/faq/bash-for-loop/

   http://www.thegeekstuff.com/2011/07/bash-for-loop-examples/