University College of London

COMP3085 / M0854

Computational Photography and Capture

Coursework 2 - Optical Flow

Author:
Pedro Moril Peñalver

Professor:
Dr. Gabriel Brostow
Dr. Tim Weyrich

March 6, 2015

Contents

Part I

Basic section

The function CW2 returns three different videos. These videos are:

1. videoAudio: This video contains the collection of frames that are on the path which flow is the nearest to the target point and the Pac-Man sound applied.
2. videoPath: This video contains the collection of frames that are on the path which flow is the nearest to the target point.
3. videoSlo: This video renders slow motion of frames visualized in videoPath.

I have encountered a number of problems trying to compute the optical flow. Firstly, the way the optical flow was presented was quite confusing. Having a file of 3.2 GB to render was quite frustrating, as you will need a machine with at least 4 GB of RAM. Secondly, the coordinates of the user's input are different that the ones used with the optical flow. That took me a lot of time to discover it.

Part II

Advance section

1 Audio interaction

In this part of the coursework, I have tried to introduce audio in an interactive way. What I have done is use the package vision included in Computer Vision System Toolbox. In Matlab, is quite complex to try to append audio into a video, but after a number of methods that I have tested, this one is the one I finally used. Using the package mentioned before, we can append the audio using "audioread" to load an wav file and videoFReader and videoFWriter to load a video previously rendered. To create the wav file, I have used Audacity. In broad strokes, the function has as an input the frames that are bounded to the final path selected to render the best match of the target points. Once we have those frames, we need to use the step function to append the audio file to each frame. This creates a video where Gabe, with every movement, produces a Pac-Man sound. The algorithm created seems quite robust and can be improved in a number of different ways. For example, using the optical flow produced by the frames, we can change the sound used according to the direction of the flow.

On the final project of the module, I will try to improve this function with the knowledge that I have now and used it to create a more complex output.

2 Slow Motion

In this case, I have used the functions provided with the coursework. These functions are WarpFL.m and WarpFLColor.m. Firstly, we have to compute all the optical flows between all the frames that belong to the path. Secondly, we have to divide those flows with a factor (in our case we use a factor of 10, and incrementing that factor by 10 until it reaches 100). What I am trying to do with the factor is divide the flow as it is a percentage. To create Slow Motion, we need to create new frames from the ones displayed. If we divide the optical flow between two frames and save that new frame, we will have a new frame to put in between the two processed frames. This will create a smooth animation between them. The more images we create, the smoothest the image will be. For instance, in our case we create 10 new images that are following the optical flow of the whole pair of frames. The problem I am encountered is that there is movement in the whole frame and the transitions are quite big. In addition, this creates black frames and is very unstable.

I have tried to find a way to fix this, but I can not reproduce the error I am having in my code. I have tried computing the optical flow between the first two nodes, put that optical flow into the WarpFLColor.m function and apply a incremental value to the source points, having in mind the optical flow too. It seems that I am having problems storaging the frames produced by the function.

The code in this part is commented as I did not have enough memory to render all the images produced on my computer.