Team 1:

Amber Straub, Anisha Aggarwal, Fiona Hebb, Jordan Lewis, Vinitha Gadiraju

Dr. Allen Malony

CIS 431/531

18 March 2018

Team 1 Project Proposal

Introduction

Since the early 2000s, computer hardware engineers have avoided the inherent power consumption and heat problems of high frequency single processors by moving towards a multiprocessor paradigm. Multicore processors are now commonplace, and industry trends continue to market four, eight, ten, and sixteen-core processors to consumers. The rapid speed at which the computer hardware industry is developing will continue to create processors with additional cores. Without new software to utilize the power of these processors, much of their benefit will remain unseen.

Project Outline

We will be implementing several parallel distortions over an .avi video using common patterns such as stencils and maps. This will include distortions over each visual time slice and audio time slice. The implementation will be coded in C++ utilizing OpenCV for the image reading, modification, and writing, with our choice of parallelization to be determined.

Background

Besides basic image processing, no one in our group has modified a video with a code implementation. Given this restraint learning the OpenCV library will be a learning curve. However the documentation of OpenCV implies it has everything we need to access the visual and audio data and implement a parallelization.

Modern data video processing always takes advantage of parallelism, because video files are so large it is necessary to do so. However for this project exploring the implementation of a parallel image distortion will give us a better idea of how exactly this efficiency is achieved.

Motivation

While the average user may not need the power of 16 cores, computationally complex tasks such as image/video editing and 3D graphics rendering can be completed efficiently with the use of multithreading and parallel computing patterns. By completing this project, we intend to demonstrate the benefits and facility of parallel computing, as well as to learn more about video/audio processing in a professional environment.

Plan

OpenCV's VideoCapture class offers the capability of retrieving time slices of visual data. This visual data will be concatenated into a buffer of images. Each image will be independent, thus we have the perfect opportunity to implement a parallel distortion on the buffer contents. We will be doing something similar with audio data, and then combining the data together again for our output .avi. We will be coding in C++. Some possible distortions include making the video black and white, increasing video contrast, speeding up or slowing down the video, etc.

Resources

OpenCV, .avi source files, Parallel libraries: TBB, OpenMP, ISPC

Finished Filters

- Invert: The invert filter flips the video so that the top is on the bottom, and the bottom is on the top. We accomplish this by replacing the color values for each pixel of an image by the color values of the pixel opposite of itself.
- Blur: The blur filter uses the algorithm for Gaussian algorithm. This algorithm uses the stencil method to average the pixel values around it. We are using a kernel size of 31.
- Black and White: This filter makes each frame the black and white version of the frame using a color luminance method.
- Negative: The negative filter inverts the colors of each pixel in each frame, one frame at a time.
- Watermark: The watermark filter overlays an image onto the video. The position of the watermark can be altered by changing the offset in warp_serial.cpp and warp_parallel.cpp. The opacity can also be changed in both of these files to change the opacity of the watermark image.
- Darken: Darken reduce image brightness with bit manipulation.
- Self Overlay: This filter makes an inverted copy of the frame and then puts it on top of the current frame creating a mirrored effect on both the top and bottom.

Results (using bird.avi)

- Invert Runtime:

Serial	Parallel
1392 ms	713 ms

- Blur Runtime:

Serial	Parallel
6128 ms	5108 ms

- Black and White Runtime:

Serial	Parallel
1464 ms	741 ms

- Negative Runtime:

Serial	Parallel
1358 ms	678 ms

- Watermark Runtime:

Serial	Parallel
1509 ms	751 ms

- Darken Runtime:

Serial	Parallel
1081 ms	852 ms

- Self Overlay Runtime:

Serial	Parallel
1914 ms	807 ms

Challenges

The hardest task of this project turned out to be getting OpenCV to run across different platforms. It could be argued that OpenCV is not very portable. However, after that set back, we found OpenCV to be not all that hard to implement code for, and we found that it had a lot of built in functions that made it possible to accomplish what we hoped to, like the ability to isolate each frame of a video separately to manipulate them. Our methods for doing this were also incredibly easy to make parallel (one would say embarrassingly). Other challenges we faced included making sure that when we parallelized our serial code, we were still getting our frames in the proper order in some cases, as well as that the frames were being manipulated properly in cases where we were using methods that involved stencil algorithms to affect the pixels in a semi-dependent manner.

The distortions struggle on larger inputs due to the large memory needed to create a buffer of frame objects. For the test file twominute.avi, there are over 3000 frames. Allocating a input and output buffer crashed our machines, granted our VM's are very lightweight. On a larger memory system, with a larger heap, this would not be a problem. As a result, the following run times reflect testing on smaller inputs, with a frame rate ranging from 200 to 500. We are happy to show a marked improvement in performance from the serial to the parallel version.

Finally, it is important to note that out virtual machines are limited to 4 processors. Given more processors it would become apparent that the program is embarrassingly parallel.

Insights

Parallel video manipulation with OpenCV turned out to be comparatively easy to the image manipulation we practiced in lab and the lab assignments. Since the manipulations required us to use for loops and pixel data is generally independent (except when we had to use stencil methods), there was a lot of chances for parallelism to be implemented through out our serial manipulations. Our challenges revolved around documentation and machine capacities.

Where to go from here

From what we've learned from this project, we'd like to go further in the kinds of manipulations and filters we can apply. There are many ideas for filters, like filters that could mimic some of the effects/styles of popular artists. We also wanted to branch out further into altering the sound of videos in parallel. That way we could combine what we've done so far with that and provide a whole suite of video editing capabilities that would run in parallel. We'd also like to develop some form of GUI for this project as it would then be more user friendly. Further, we were interested in seeing if we could determine the best way of parallelizing these filters using other parallelization techniques we've learned and used in class. One idea was to try parallelizing the serial code using ISPC or TBB rather than OpenMP. Overall, this project increasingly broadened our knowledge of how to apply parallel techniques and what can be done using parallel computing.

Project responsibilities

Vinitha:

Watermark implementation, testing all filters in linux, parallelize filters with OpenMP, QA, ~~GUI~~

Jordan:

~~Sepia~~(work in progress), B&W, ~~Twirl Distortion~~(in progress)

Amber:

Serial implementation of OpenCV object, inversion, darken, overlay, ~~encrypt/decrypt~~ (in progress), parallel buffers and heap allocation, QA

Anisha:

Blur utilizing stencil method, Negative distortion, final report

Fiona:

~~Andy Warhol Filter~~ (in progress), final report

Initial Meeting Notes

*Component completed

To Do:

*Get .avi test files

*Install openCV libraries

*Can we isolate the first image of the vid and save it?

*Can we do this for all images?

*Can we access image pixels and alter them?

*Can we write to an image and write back changes to vid?

Can we isolate audio pieces? nope

Can we alter audio and write it back? nope

Once these work:

Implement parallelization

*What library do we want to use? I like openMP

*I want to apply a video watermark - vinitha

*Anyone want to do a stencil blur? -anisha

Other video editing ideas? Everyone should implement 1 hard or two medium edits, then from main file you can insert a video and then select your output type.

Someone is welcome to do this in a GUI, or we can do it command line(GUI can be your portion of the project?)

Sfml - grid based GUI