CSCI 347 - Project 3 Report

Preston Duffield
Western Washington University
May 28, 2023

Edge Detector

Edge Detector is an edge detection program that uses a laplacian filter to highlight the edges in a set of images. The images are specified as command line arguments and the project is designed to process multiple images concurrently using threads. The goal of the project is to concurrently apply the laplacian filter to multiple images, improving the execution time compared to sequential execution.

The program contains several functions, structs, and a global variable, all of which work together to process the images:

1. `PPMPixel` Struct: This struct represents a pixel in an image, with the RGB values of the pixel.
2. `parameter` Struct: This struct holds the information each thread needs to perform its task.
3. `file_name_args` Struct: This struct is used to hold the input and output filenames for each image.
4. `total_elapsed_time` Global Variable: This is a double value that tracks the total time taken by all threads to process all input images.
5. `compute_laplacian_threadfn`: This is the thread function. It applies the Laplacian filter to a section of the image.
6. `apply_filters`: This function uses threads to apply the Laplacian filter to an image and measures the time taken.
7. `write_image`: This function saves a processed image to a file.
8. `read_image`: This function reads an image file and returns a PPMPixel pointer to the pixel data.
9. `manage_image_file`: This is another thread function. It manages the entire process of reading an image file, processing the image, and writing the result to a file.
10. `main`: This is the entry point of the program. It checks command line arguments, creates threads to process each image file, and prints the total elapsed time.

Concurrency

Concurrency in this program is achieved through the use of POSIX threads (pthreads). For each image file passed as a command line argument, a new thread is created in the main function. Each of these threads runs the `manage_image_file` function, which manages the processing of one image.

Within the `manage_image_file` function, the `apply_filters` function is called, which further divides the work among a specified number of threads (defined by the `LAPLACIAN_THREADS` macro). Each of these threads runs the `compute_laplacian_threadfn` function, which applies the Laplacian filter to a section of the image.

In terms of handling possible race conditions, the main area where this could occur is when updating the `total_elapsed_time` global variable. However, as each image file is processed by a separate thread and the time taken to process each image is added to `total_elapsed_time` in a single operation, there's no risk of

race conditions here as the operation is atomic. Furthermore, as the processed images are written to separate files, there is no risk of a race condition when writing the output.

Experiments

Results

Implications of Results

Conclusion

Appendix