Team: Swaraj Kaondal – skaonda@ncsu.edu, Vaibhavi Shetty – vshetty2@ncsu.edu

Parallelizing Canny Edge Detection using MPI and OpenMP

Problem Statement

Edge detection is a fundamental step in image processing and computer vision applications. The Canny edge detection algorithm is widely used due to its ability to detect edges efficiently while reducing noise. However, its computational cost makes it a performance bottleneck, especially for large images. To address this, we aim to parallelize the Canny edge detection algorithm using MPI and OpenMP, leveraging distributed and shared memory parallelism to enhance performance.

Canny Edge Detection Algorithm Steps

The Canny edge detection algorithm consists of the following steps:

1. **Gaussian Blur** – Smooth the image to reduce noise using a Gaussian filter.
2. **Gradient Calculation** – Compute the intensity gradient using Sobel operators.
3. **Non-Maximum Suppression** – Thin out the edges by keeping only the local maxima in the gradient direction.
4. **Double Thresholding** – Classify edges into strong, weak, and non-relevant edges.
5. **Edge Tracking by Hysteresis** – Finalize edge detection by connecting weak edges to strong edges if they are connected to form continuous contours.

Proposed Methods

We will implement the Canny edge detection algorithm in a parallelized manner using:

1. **MPI (Message Passing Interface)** – For distributed parallelization, dividing the image across multiple nodes and aggregating the results.
2. **OpenMP (Open Multi-Processing)** – For shared-memory parallelization, utilizing multi-threading to optimize computation within a single node.

At the end of the project, we will compare the execution time, scalability, and efficiency of all three approaches.

Milestones

1. **Week 1-2**: Implement the sequential Canny edge detection algorithm and validate its correctness.
2. **Week 3-4**: Implement the MPI-based parallelization and evaluate its performance on a distributed system.
3. **Week 5-6**: Implement the OpenMP version and compare it with MPI.
4. **Week 7**: If time permits, implement the CUDA version for GPU acceleration.
5. **Week 8**: Conduct performance analysis and comparisons, documenting results and conclusions.

By the end of the project, we will have a comprehensive performance comparison of different parallelization approaches for Canny edge detection.