Project - The Hough transform

I. Goal

Implement the Hough transform and create a multithreaded and a distributed version in MPI. Compare the execution times.

II. Introduction

The Hough transform is a technique used in image analysis to detect lines present in the image based on the output of an edge detection algorithm. The edge detection can be done in any manner, whether it is Canny edge detection or other similar algorithms. The output of the edge detection algorithm is fed into the Hough transform algorithm, which draws the detected lines on the image.

III. Theory

In general, the straight line $y = mx + b$ can be represented as a point (b, m) in the parameter space. However, vertical lines pose a problem. They would give rise to unbounded values of the slope parameter m. For this reason, the line is written in its polar coordinate form

$r = x\cos(\theta) + y\sin(\theta)$

where r is the distance from the origin to the closest point on the straight line, and θ is the angle between the x axis and the line connecting the origin with that closest point. It is therefore possible to associate with each line of the image a pair (r, θ).

The linear Hough transform algorithm estimates the two parameters that define a straight line. The transform space has two dimensions, and every point in the transform space is used as an accumulator to detect or identify a line described by $r = x\cos(\theta) + y\sin(\theta)$. Every point in the detected edges in the image contributes to the accumulators.

IV. Implementation

A. Parallel implementation

$To\ be\ continued$

B. Distributed implementation

The distributed Hough transform has been implemented with MPI (Message Passing Interface), by splitting the work between 3 worker processes equally. In total, there are 4 processes, the master and 3 workers.

Only the voting part of the algorithm is distributed.

V. Results

At first glance, the distributed version seems to be faster than the sequential one.

Image	Sequential	Parallel	Distributed
sudoku_pencil_2.jpg	0.592	-	0.369

TABLE I: Table with execution times (seconds)