OS Laboratory - 5
Report

Name: Hemanth Reddy **Roll no:** 180010023

1. **Image Transformations:**

   Two transformation were used :

   a. **Grayscale Conversion (T1)**

      Here we convert a color image into a grayscale image, by converting RGB values of a pixel to a single grayscale value.

      gray = 0.3 * red + 0.59 * green + 0.11 * blue;

   b. **Edge Detection (T2)**

      Here we detect edges by taking the grayscale image as input. By convolving the image matrix with a filter(taken stride=1,w=3,h=3) we can get the edge detected image.

2. **Method used to prove pixel correctness**

   By using the command diff provided in linux, we can find the difference between two files i.e.. whether the two files are similar or not!

   Motivation : For each of the methods(in parts 1,2_1a,2_1b,..) the output ppm file after doing grayscale and edge detection remains the same.

   Command : diff <out_img1.ppm> <out_img2.ppm>

3. **Run-time and speed-up**

Part	**Time Taken (s)**
Part 1	0.262
Part 2_1a	0.247
Part 2_1b	0.246
Part 2_2	0.222
Part 2_3	0.267

Part 1 is taking the highest time among all. It is because of sequential execution of transformations.

Part 2 : , here there is a sharing of transformed data between both threads.So, they execute a bit faster than previous.

Part 2 : 2, Here both processes use shared memory, where, when one process writes something in the memory then it can be read by the other process at same time.

Part 2 : 3, Here we use pipes,this is slower than a shared memory situation because to read, the process has to wait till the pipe becomes full.

4. **Ease of Each Approach**

Part 1:

The part 1 is easy of all the other parts and subparts.Here we just had to read the image and perform calculations on the input matrix.

Part 2_1a & Part 2_1b :

It was easy to implement, Here we just need to be careful while adding lines of code like semaphores/locks or while loops and also the main problem is we should carefully pass the arguments of functions since this section involves many function calls.

Part 2_2:

This part is a hard one, as I had to create a shared memory, and help processes communicate through that.It involved many segmentation faults which were cleared by passing integers through character arrays(check code).

Part 2_3:

This process is also a hard one, Here we need to create a pipe and consider all the cases like when the pipe becomes empty,full,....We parallelize it by sending each row from the *grayScale* function to the *EdgeDetection* function when it's done getting values.

Output Image :
(Chess Board is the Input)