Assignment 2: Parallel Image filtering using MPI

Digital image filters are prevalent today and are widely used from enhancing images on image sharing platforms to detecting edges in computer vision. Let's implement one such filtering parallelly using MPI.

If you are not familiar with how image filtering works, know more about it here , here and here.

Objective

Your task is simple. You will be given MxN matrix of pixel values and a KxK filter/kernel (k is odd). For each element in the matrix, align this element with the center of the kernel (there will always be a center, as kernel dimensions are odd), perform convolution operation (as described here) to get new pixel value. When you align filter center with any pixels around the edge of the image, part of the filter will hang outside the image boundary. Handle this by wrapping around the pixel values (as described here). When calculating new pixel values, you will only consider original pixel values and not the updated values. Files containing sample input and output are attached with this pdf.

Input

- Read from standard input stream
- First line of input contains 3 space separated integers M, N, K. M and N are the number of rows and columns of the input matrix respectively. K (i.e. KxK) is the dimension of the filter/kernel.
- Next M lines follows. Each line contains N space separated **integers** describing a row of the matrix.
- Next K lines follows. Each line contains K space separated **floats** describing a row of the filter.

Output

- Write to standard output stream
- M lines, each line is a row of the matrix after filter is applied
- Each line will contain N space separated **integers** describing each line of the filtered matrix
- After performing convolution, round it off to nearest integer before outputting it.

Submission

- A single c/c++ file named *roll_no*.c (ex. 14256.c)
- A makefile to build executable. Name of the generated executable must be *roll_no* (ex. 14256)
- A readme file describing your parallelization approach in few sentences
- Compress files in a zip named assn2_*roll_no*.zip (ex. assn2_14256.zip)

Marks Distribution (Out of 100)

- Proper indented and well commented code: 10 marks
- Parallelization approach: 25 marks
- Test cases: 65 marks

Important Notes

- Your code will be run with **mpirun**, and number of processors will be given as parameter.
- There will be **automated** test case evaluation, so your output and submission file format **must** exactly match the format specified, otherwise you will get a 0 in test case component
- C/C++ are the only allowed languages. You are only allowed to use MPI as a communication mechanism between processes, you can't use files to communicate between processes.
- You will get a 0 if your approach is not actually parallel, even if all the testcases passes
- Zero tolerance to plagiarism. You are not allowed to **copy** any snippet from anywhere. If you have used some algorithm from anywhere, give a reference.