Xi'an Jiaotong-Liverpool University
西交利物浦大學

XJTLU Entrepreneur College (Taicang) Cover Sheet

Module code and Title	**DTS202TC Foundation of Parallel Computing**
School Title	**School of AI and Advanced Computing**
Assignment Title	**Group Assessment 1**
Submission Deadline	**Sunday, Nov 19th,2023 @ 23:59**
Final Word Count	**N/A**
If you agree to let the university use your work anonymously for teaching and learning purposes, please type **"yes"** here.	

I certify that I have read and understood the University's Policy for dealing with Plagiarism, Collusion and the Fabrication of Data (available on Learning Mall Online). With reference to this policy I certify that:

- My work does not contain any instances of plagiarism and/or collusion.
  My work does not contain any fabricated data.

By uploading my assignment onto Learning Mall Online, I formally declare that all of the above information is true to the best of my knowledge and belief.

Scoring – For Tutor Use	
Student ID	

Stage of Marking	**Marker Code**	**Learning Outcomes Achieved (F/P/M/D) (please modify as appropriate)**			**Final Score**
		A	**B**	**C**	
1st Marker – red pen					
Moderation – green pen	**IM Initials**	The original mark has been accepted by the moderator (please circle as appropriate):			Y / N
		Data entry and score calculation have been checked by another tutor (please circle):			Y
2nd Marker if needed – green pen					

For Academic Office Use			**Possible Academic Infringement (please tick as appropriate)**	
Date Received	**Days late**	**Late Penalty**	☐ **Category A**	Total Academic Infringement Penalty (A,B, C, D, E, Please modify where necessary) _____
			☐ **Category B**	
			☐ **Category C**	
			☐ **Category D**	
			☐ **Category E**	

DTS202TC Foundation of Parallel Computing

Group Assignment 1

Due: Sunday Nov. 19th, 2023 @ 11:59pm
Weight: 30%
Maximum score: 100 points (70 for group work + 30 for peer assessment)

Overview

The purpose of this assignment is to gain experience in C programming and parallel algorithm design. You are expected to write a C serial program to rotate a grayscale image (without parallelism), and provide a parallel algorithm design that speeds up the serial implementation. Identify the bottlenecks in your serial implementation and provide an outline of how you plan to parallelise the serial implementation. Highlight the performance improvements you expect after parallelisation and potential challenges you may have to overcome. You may include pseudo-code or explain your approach in plain English.

Learning Outcomes

- A. Identify serial and parallel algorithm.
- B. Appreciate basic principal and techniques in devising parallel algorithm.
- I. Apply common parallel algorithm patterns.
- K. Identify and solve a computational problem with parallel algorithm design and program.

Team policy

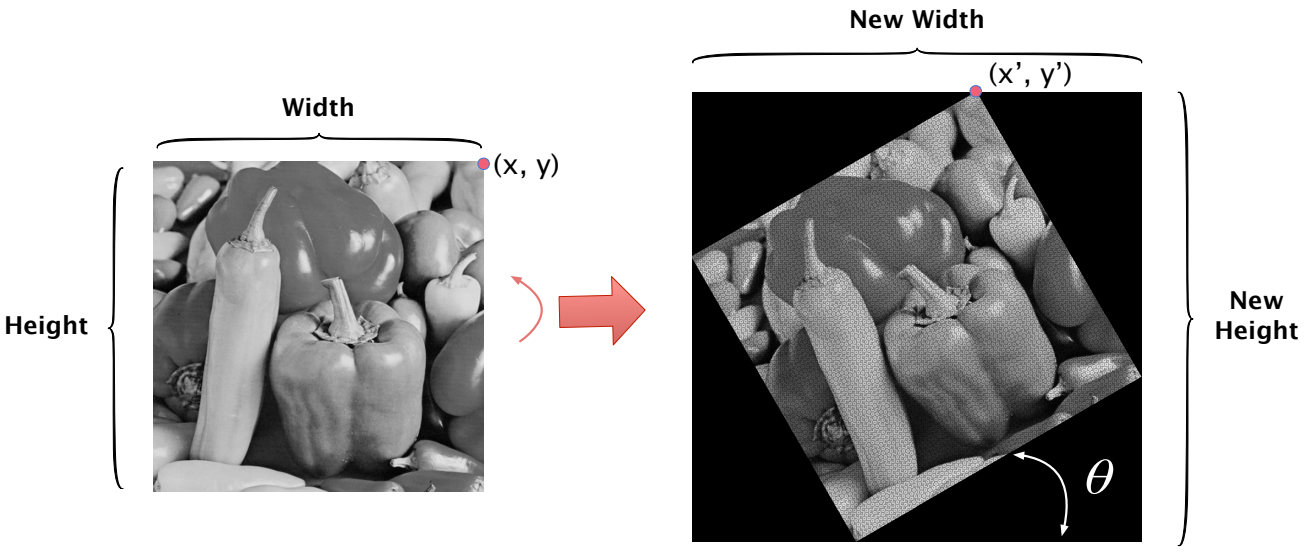
You are free to form your group up to 5 team members (with a minimum of two members), you need to submit all team members' information before 13th Nov, 23:59 via LMO. Students who fail to do so will be randomly assigned to a team. Changes will not be allowed once the teams have been confirmed.

Avoid Plagiarism

- Do ***not*** submit work from other teams.
- Do ***not*** share code/work to students other than your own team members.

- Do ***not*** read code/work from other teams, discussions between teams should be limited to high level only.
- Do ***not*** use open-source code.

Tasks



Algorithm 1

$new\_height \leftarrow | height * cosine(\theta) | + | width * sine(\theta) |$
$new\_width \leftarrow | width * cosine(\theta) | + | height * sine(\theta) |$ ▷ Define the height and width of the new image rotated

$ori\_centre_w \leftarrow width/2$
$ori\_centre\_h \leftarrow height/2$
$new\_centre\_w \leftarrow new\_width/2$
$new\_centre\_h \leftarrow new\_height/2$
for $i = 0, 1, 2, \ldots, height - 1$ **do**
    for $j = 1, 2, \ldots, width - 1$ **do**
        $x \leftarrow width - ori\_centre\_w - j$
        $y \leftarrow height - ori\_centre\_h - i$ ▷ coordinates of pixel with respect to the centre of original image

        $new\_x \leftarrow x * cosine(\theta) + y * sine(\theta)$
        $new\_y \leftarrow y * cosine(\theta) - x * sine(\theta)$
        $new\_x \leftarrow new\_centre\_w - new\_x - 1$
        $new\_y \leftarrow new\_centre\_h - new\_y - 1$ ▷ coordinates of pixel with respect to the centre of new image
        $img\_out[new\_y][new\_x] \leftarrow img\_in[i][j]$ ▷ make sure new_x and new_y are integers, and don't forget to check the boundaries
    end for
end for

1 Serial Version (30 points)

1.1 Image Rotation

Image rotation is one of the most common image processing algorithms. To complete the rotation we need to write the pixel value for each pixel in the new location. Using some High School geometry we can work out the relationship between the source coordinates (x, y) and the destination coordinates (x', y')

$$\begin{aligned} x' &= x\cos\theta + y\sin\theta \\ y' &= -x\sin\theta + y\cos\theta \end{aligned} \tag{1}$$

A complete pseudocode has been provide in Algorithm 1.

You will see some dots after this simple rotation algorithm, don't worry about that for this coursework.

1.2 Reading and Writing Image

You will need to read a PGM image, process the rotation, then write the rotated PGM image back to the file system. **Plain PGM** format is a simple grayscale graphic image format, each pixel is represented by its grey value number, with 0 being black and Maxval (defined in the PGM file) being white. The below image.pgm is given as an example, more detail pgm specifications can be found at http://davis.lbl.gov/Manuals/NETPBM/doc/pgm.html

image.pgm

```
P2
24 7
15
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0 15  0
0  3  3  3  0  0  0  7  7  7  0  0  0 11 11 11  0  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0  0  0
0  3  0  0  0  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Your program should be compiled and executed by:

```
make
./rotate_image {degree in int}
```

Your program read the im.pgm file and output an rotated image named im-rotated.pgm

2 Parallel Algorithm Design (30 points)

Now that you have a complete understanding of the task, do the following:

- Identify and analyse the bottlenecks of your serial implementation using necessary profiling tools. (**10 points**), and

- provide a design of a solution to speed up using parallel programming (**20 points**).

You do not need to do the actual coding for the parallel implementation, but only provide the detailed design with maximum 500 words. You may include pseudo-code or explain your approach in plain English. Highlight the performance improvements you expect after parallelisation and potential challenges you may have to overcome.

3 Peer Assessment (30 points)

Please review your peers based on the actual contributions. This will be done on LMO anonymously, each of the group members should login their LMO account and submit the marks individually. Marks should be submitted as soon as the group work submission is done. Peer review rubrics are attached in the appendix table.

4 Submission (10 points)

One of the group members must submit the following files:

- ***rotate-image.c*** C code of the serial implementation .
- A ***Makefile*** that will compile your code, make sure the output executable names are correct.
- A ***report.pdf*** file contains all the source code and the parallel design. You should also include the ***Cover Page*** with the student ID of all group members (template can be found on LMO) in the first page of your pdf.

Once you have all the files, please put them in a single directory (named groupid-A1) and compress them into to a single ***.zip*** file. You must follow the following structure:

```
|—— {groupid}_A1.zip
|      |—— report.pdf
|      |—— code
|      |      |—— rotate-image.c
|      |      |—— Makefile
```

The assignment must be submitted via Learning Mall to the correct drop box. Only electronic submission is accepted and no hard copy submission. All students must download their file and check that it is viewable after submission. Documents may become corrupted during the uploading process (e.g. due to slow internet connections). However, students themselves are responsible for submitting a functional and correct file for assessments.

Please note that quality of report and correctness of submission will also be marked (**10 points in total**, 5 points for the quality of report, 5 points for the correctness of submission).

Table 1: Peer Review Rubrics

Marks	10	7	4	0
Contributions	Routinely provides useful ideas when participating in the group discussion. A leader who contributes a lot of effort.	Usually provides useful ideas when participating in the group discussion. A strong group member who tries hard!	Sometimes provides useful ideas when participating in the group discussion. A satisfactory group member who does what is required.	Rarely provides useful ideas when participating in the group discussion. May refuse to participate.
Problem-solving	Actively looks for and suggests solutions to problems.	Refines solutions suggested by others.	Does not suggest or refine solutions, but is willing to try out solutions suggested by others.	Does not try to solve problems or help others solve problems. Lets others do the work.
Working with others	Almost always listens to, shares with, and supports the efforts of others. Tries to keep people working well together.	Usually listens to, shares, with, and supports the efforts of others. Does not cause "waves" in the group.	Often listens to, shares with, and supports the efforts of others, but sometimes is not a good team member.	Rarely listens to, shares with, and supports the efforts of others. Often is not a good team player.

Category	Exemplary (8-10)	Accomplished (6-7)	Capable (4-5)	Need Improvement (0-3)
Functionality (Correctness and output)	Code produces correct output. Demonstrate effective use of programming features as well as functions and libraries. Code is optimized for the maximum performance.	Code produces partially correct output. Partially missing code or minor issues with excellent logic. Performance is not optimal.	Code doesn't compile. However, design idea is clear. Shows some understanding of the tasks and uses relevant programming features	Code doesn't compile. Design idea is not obvious. Doesn't know much understanding of the task.
Quality (Structure and best practice)	Excellent code quality with correct use of function and programming logic. Excellent design and the codes are commented, or design idea is clearly explained.	Code compiles. Result is faulty. Design and structure are not intuitive. Inappropriate use of function or algorithms.	Code doesn't compile. Incomplete code. However, design and structure are intuitive.	Code doesn't compile. Incomplete code. Design and structure are incorrect.
Identify and analyze of bottleneck	Correct identification of performance bottleneck, demonstrated evidence of correct usage of tools, demonstrated correct and comprehensive justifications and analysis.	Correct identification of performance bottleneck, demonstrated evidence of usage of tools with moderate justifications and analysis.	Mostly correct identification of performance bottleneck, limited demonstration of usage of tools, limited justification and analysis.	Limited attempt without any evidence of usage of tools, nor justifications.
Parallel algorithm design	Provide a comprehensive design of parallel algorithm that is logically correct, easy to understand and well written. Provide the estimated performance improvement with comprehensive justification aligned with the algorithm design.	Provide a design of parallel algorithm design with sufficient justifications and details. Some critical points are missing or contains obvious mistakes.	Provide a design of parallel algorithm design with limited justifications and details. Contains critical mistakes.	Limited attempt without any justifications or points that are not relevant to the questions.