NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA SURATHKAL
DEPARTMENT OF INFORMATION TECHNOLOGY
IT 301 Parallel Computing LAB 6

P Akshara - 181IT132

1. Write a parallel program (using Openmp) to convert a color image to grayscale and YIQ. The RGB values (in decimal) are already extracted and stored in "KittenRGB.txt" file. Read the input values from the file.
a. Compute the grayscale conversion using luminosity method, that is, G=R*0.21+G*0.72+B*0.07.
b. Here is the RGB -> YIQ conversion:
Y = 0.299*R + 0.587*G + 0.114*B
I = 0.596*R – 0.275*G – 0.321*B
Q = 0.212*R – 0.523*G + 0.311*B
Analysis: Compare the time taken for the computation with Single thread and Multiple threads(2,4,8,16) . Prove that parallel computation is faster than serial.

```
#include <bits/stdc++.h>
#include <omp.h>
using namespace std;
#define N 300
int mat[N * N * 3], mat_gray[N * N], mat_yiq[3 * N * N];
int main()
{
freopen("KittenRGB.txt", "r", stdin);

// Storing pixel values in a 300*300 matrix
for (int i = 0; i < 3 * N * N; i++)
{
cin >> mat[i];
}
// sequential code
double start, end;
// rgb to gray scale
start = omp_get_wtime();
for (int i = 0; i < 300 * 300 * 3; i += 3)
{
int R = mat[i];
int G = mat[i + 1];
int B = mat[i + 2];
mat_gray[i] = (R * 0.21) + (G * 0.72) + (B * 0.07);
mat_yiq[i] = (0.299 * R) + (0.587 * G) + (0.114 * B);
mat_yiq[i + 1] = (0.596 * R) - (0.275 * G) - (0.321 * B);
mat_yiq[i + 2] = (0.212 * R) - (0.523 * G) + (0.311 * B);
}
end = omp_get_wtime();
cout << "Time taken by sequential code = " << (end - start) << "s\n";
// parallel
// num_threads=2
start = omp_get_wtime();
#pragma omp parallel num_threads(2)
{
#pragma omp for
for (int i = 0; i < N * N * 3; i += 3)
{
int R = mat[i];
int G = mat[i + 1];
int B = mat[i + 2];
mat_gray[i] = (R * 0.21) + (G * 0.72) + (B * 0.07);
mat_yiq[i] = (0.299 * R) + (0.587 * G) + (0.114 * B);
```

```
mat_yiq[i + 1] = (0.596 * R) - (0.275 * G) - (0.321 * B);
mat_yiq[i + 2] = (0.212 * R) - (0.523 * G) + (0.311 * B);
}
}
end = omp_get_wtime();
cout << "Time taken by 2 threads in parallel = " << (end - start) << "s\n";
// num_threads=4
start = omp_get_wtime();
#pragma omp parallel num_threads(4)
{
#pragma omp for
for (int i = 0; i < N * N * 3; i += 3)
{
int R = mat[i];
int G = mat[i + 1];
int B = mat[i + 2];
mat_gray[i] = (R * 0.21) + (G * 0.72) + (B * 0.07);
mat_yiq[i] = (0.299 * R) + (0.587 * G) + (0.114 * B);
mat_yiq[i + 1] = (0.596 * R) - (0.275 * G) - (0.321 * B);
mat_yiq[i + 2] = (0.212 * R) - (0.523 * G) + (0.311 * B);
}
}
end = omp_get_wtime();
cout << "Time taken by 4 threads in parallel = " << (end - start) << "s\n";
// num_threads=8
start = omp_get_wtime();
#pragma omp parallel num_threads(8)
{
#pragma omp for
for (int i = 0; i < N * N * 3; i += 3)
{
int R = mat[i];
int G = mat[i + 1];
int B = mat[i + 2];
mat_gray[i] = (R * 0.21) + (G * 0.72) + (B * 0.07);
mat_yiq[i] = (0.299 * R) + (0.587 * G) + (0.114 * B);
mat_yiq[i + 1] = (0.596 * R) - (0.275 * G) - (0.321 * B);
mat_yiq[i + 2] = (0.212 * R) - (0.523 * G) + (0.311 * B);
}
}
end = omp_get_wtime();
cout << "Time taken by 8 threads in parallel = " << (end - start) << "s\n";
// num_threads=16
start = omp_get_wtime();
#pragma omp parallel num_threads(16)
{
#pragma omp for
for (int i = 0; i < N * N * 3; i += 3)
{
int R = mat[i];
int G = mat[i + 1];
int B = mat[i + 2];
mat_gray[i] = (R * 0.21) + (G * 0.72) + (B * 0.07);
mat_yiq[i] = (0.299 * R) + (0.587 * G) + (0.114 * B);
mat_yiq[i + 1] = (0.596 * R) - (0.275 * G) - (0.321 * B);
mat_yiq[i + 2] = (0.212 * R) - (0.523 * G) + (0.311 * B);
}
}
end = omp_get_wtime();
cout << "Time taken by 16 threads in parallel = " << (end - start) << "s\n";
// write grayscale to file
freopen("GrayScale.txt", "w", stdout);
cout << "Gray Scale format\n";
```

```
cout<<"-------------------------------\n";
for (int i = 0; i < N * N; i++)
cout << mat_gray[i] << " ";
// write yiq to file
freopen("YIQ format.txt", "w", stdout);
cout << "YIQ format\n";
cout<<"-------------------------------\n";
for (int i = 0; i < N * N * 3; i++)
cout << mat_yiq[i] << " ";
}
```

Output

```
(base) akshara@akshara-VivoBook-ASUSLaptop-X530FN-S530FN:/media/akshara/DATA/NITK/Lab-Sem5/IT301 PC/Lab 6$ g++ -fopenmp lab6.cpp
(base) akshara@akshara-VivoBook-ASUSLaptop-X530FN-S530FN:/media/akshara/DATA/NITK/Lab-Sem5/IT301 PC/Lab 6$ ./a.out
Time taken by sequential code = 0.00285413s
Time taken by 2 threads in parallel = 0.00129632s
Time taken by 4 threads in parallel = 0.000684374s
Time taken by 8 threads in parallel = 0.00130181s
Time taken by 16 threads in parallel = 0.00118431s
```

Analysis

There are approx 300*300*3 computations. Over multiple executions of the program it is shown that parallelized versions take lesser time than sequential, by distributing the iterations among several threads (2, 4, 8, 16), and due to the trade-off in time for spawning the threads vs computation time, we observe that execution is most optimal when around 4 threads are used.