

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA SURATHKAL

DEPARTMENT OF INFORMATION TECHNOLOGY

IT 301 Parallel Computing LAB 6

16th September 2020

Faculty: Dr. Geetha V and Mrs. Thanmayee

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. NOTE: The RGB.txt file has RGB values in single line and not matrix format. Ex: RGBRGB ...

The image used for extracting RGB values is 300 * 300 pixels jpg image



Code:

```
#include<bits/stdc++.h>
#include<omp.h>
using namespace std;
int main()
{
    ifstream ptr;
    int R,G,B,i;
    double Gray;
    ptr.open("KittenRGB.txt");
    double t1,t2;
    vector<vector<int>> v;
    vector<int> m;
    int size=300*300;
    vector<double>Y(size,0),I(size,0),Q(size,0),Gr(size,0);
    for(int i=0;i<size;i++)
    {
        m.clear();
        ptr>>R;
        ptr>>G;
        ptr>>B;
        m.push_back(R);
        m.push_back(G);
        m.push_back(B);
        v.push_back(m);
    }
    ptr.close();
    printf("1.serial execution\n");
    t1=omp_get_wtime();
    for(i=0;i<size;i++)
    {
        Gr[i]=(v[i][0]*0.21+v[i][1]*0.72+v[i][2]*0.07);
        Y[i]=(0.299*v[i][0]+0.587*v[i][1] + 0.114*v[i][2]);
        I[i]=(0.596*v[i][0]-0.275*v[i][1]-0.321*v[i][2]);
        Q[i]=(0.212*v[i][0]-0.523*v[i][1]+0.311*v[i][2]);
    }
    t2=omp_get_wtime();
    printf("Time taken =%lf s\n\n",t2-t1);printf("2.parallel execution with 2 thread\n");
    t1=omp_get_wtime();
    #pragma omp parallel num_threads(2)
    #pragma omp for private(i) schedule(guided,100)
    for(i=0;i<size;i++)
    {
```

```

Gr[i]=(v[i][0]*0.21+v[i][1]*0.72+v[i][2]*0.07);
Y[i]=(0.299*v[i][0]+0.587*v[i][1] + 0.114*v[i][2]);
I[i]=(0.596*v[i][0]-0.275*v[i][1]-0.321*v[i][2]);
Q[i]=(0.212*v[i][0]-0.523*v[i][1]+0.311*v[i][2]);
}
t2=omp_get_wtime();
printf("Time taken =%lf s\n\n",t2-t1);
printf("3.parallel execution with 4 thread\n");
t1=omp_get_wtime();
#pragma omp parallel num_threads(4)
#pragma omp for private(i) schedule(guided,100)
for(i=0;i<size;i++)
{
Gr[i]=(v[i][0]*0.21+v[i][1]*0.72+v[i][2]*0.07);
Y[i]=(0.299*v[i][0]+0.587*v[i][1] + 0.114*v[i][2]);
I[i]=(0.596*v[i][0]-0.275*v[i][1]-0.321*v[i][2]);
Q[i]=(0.212*v[i][0]-0.523*v[i][1]+0.311*v[i][2]);
}
t2=omp_get_wtime();
printf("Time taken =%lf s\n\n",t2-t1);
printf("4.parallel execution with 8 thread\n");
t1=omp_get_wtime();
#pragma omp parallel num_threads(8)
#pragma omp for private(i) schedule(guided,100)
for(i=0;i<size;i++)
{
Gr[i]=(v[i][0]*0.21+v[i][1]*0.72+v[i][2]*0.07);
Y[i]=(0.299*v[i][0]+0.587*v[i][1] + 0.114*v[i][2]);
I[i]=(0.596*v[i][0]-0.275*v[i][1]-0.321*v[i][2]);Q[i]=(0.212*v[i][0]-0.523*v[i][1]+0.311*v[i][2]);
}
t2=omp_get_wtime();
printf("Time taken =%lf s\n\n",t2-t1);
printf("5.parallel execution with 16 thread\n");
t1=omp_get_wtime();
#pragma omp parallel num_threads(16)
#pragma omp for private(i) schedule(guided,100)
for(i=0;i<size;i++)
{
Gr[i]=(v[i][0]*0.21+v[i][1]*0.72+v[i][2]*0.07);
Y[i]=(0.299*v[i][0]+0.587*v[i][1] + 0.114*v[i][2]);
I[i]=(0.596*v[i][0]-0.275*v[i][1]-0.321*v[i][2]);
Q[i]=(0.212*v[i][0]-0.523*v[i][1]+0.311*v[i][2]);
}
t2=omp_get_wtime();

```

```

printf("Time taken =%lf s\n\n",t2-t1);
return 0;
}

```

Output:

```

0, 32), (129, 158, 50), (65, 92, 0), (62, 88, 0), (125, 151, 44), (136, 159, 53),
, (157, 180, 74), (107, 137, 23), (84, 113, 3), (147, 180, 73), (173, 208, 106),
(73, 110, 15), (59, 99, 10), (20, 62, 0), (55, 98, 16), (77, 120, 38), (39, 81,
0), (39, 77, 0), (81, 116, 24), (91, 123, 26), (97, 123, 22), (109, 134, 30), (
166, 189, 83), (167, 208, 102), (107, 149, 41), (91, 130, 21), (106, 144, 33), (
78, 112, 0), (102, 135, 18), (130, 159, 41), (121, 149, 28), (101, 127, 2), (111
, 136, 9), (122, 147, 19), (128, 153, 25)]
jacky@jacky-Strix-G531GT-G531GT:~/pnclab$ ls
1_1.cpp                                181IT211_IT302_P3_Output_TC6.txt
181IT209_IT211_P3_Output_TC1.txt      1.cpp
181IT209_IT302_P3_Output_TC1.txt      a.out
181IT211_IT302_P1.cpp                 a.txt
181IT211_IT302_P2_Output_TC1.txt      bsearch.c
181IT211_IT302_P3.cpp                 hello
181IT211_IT302_P3_Output_TC1.txt      Kitten.jpg
181IT211_IT302_P3_Output_TC2.txt      KittenRGB.txt
181IT211_IT302_P3_Output_TC3.txt      lab6.cpp
181IT211_IT302_P3_Output_TC4.txt      para.c
181IT211_IT302_P3_Output_TC5.txt      rgb2.py

```

After running rgb2.py file, which uses Python Imaging Library, All the points corresponding to rgb for given image(Kitten.jpg) are found and KittenRGB.txt file is generated.

```

jacky@jacky-Strix-G531GT-G531GT:~/pnclab$ g++ -o hello -fopenmp lab6.cpp
jacky@jacky-Strix-G531GT-G531GT:~/pnclab$ ./hello
1.serial execution
Time taken =0.004567 s

2.parallel execution with 2 thread
Time taken =0.002481 s

3.parallel execution with 4 thread
Time taken =0.001437 s

4.parallel execution with 8 thread
Time taken =0.001255 s

5.parallel execution with 16 thread
Time taken =0.001566 s

```

Here, we observe that the time taken for serial execution is more than time taken in parallel execution. So parallel execution is faster than serial execution.

Name: BHAJAN KUMAR BARMAN,
Roll: 181IT211