DEPARTMENT OF INFORMATION TECHNOLOGY, NITK SURATHKAL

IT301 Parallel Programming

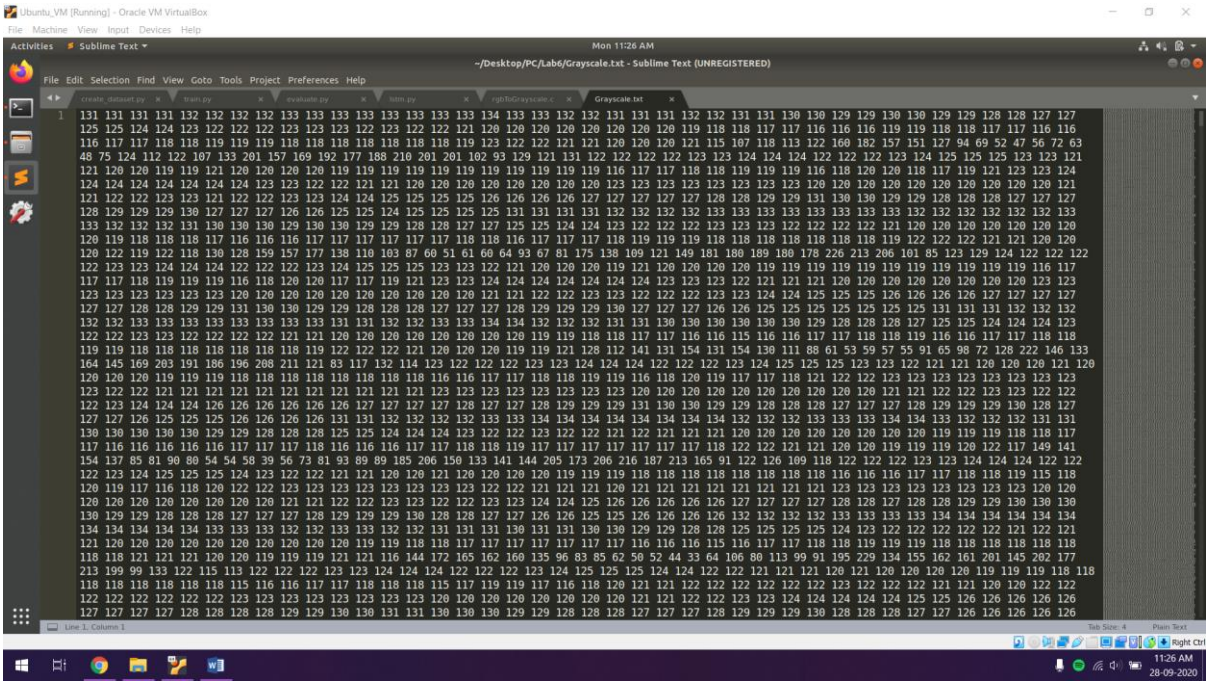
LAB 6 (16th September 2020)

Submitted by: Ajay Bharadwaj (181IT103)

1. RGB to Grayscale:

```
File  Edit  View  Search  Terminal  Help
user@user-VirtualBox:~/Desktop/PC/Lab6$ gcc -o rgbToGrayscale -fopenmp rgbToGrayscale.c
user@user-VirtualBox:~/Desktop/PC/Lab6$ ./rgbToGrayscale
Enter the name of the file containing the pixel values: KittenRGB.txt
```

```
File  Edit  View  Search  Terminal  Help
Grayscale to Output: 106
Grayscale to Output: 97
Grayscale to Output: 95
Grayscale to Output: 82
Grayscale to Output: 137
Grayscale to Output: 102
Grayscale to Output: 126
Grayscale to Output: 144
Grayscale to Output: 80
Grayscale to Output: 76
Grayscale to Output: 137
Grayscale to Output: 146
Grayscale to Output: 167
Grayscale to Output: 122
Grayscale to Output: 99
Grayscale to Output: 165
Grayscale to Output: 193
Grayscale to Output: 95
Grayscale to Output: 84
Grayscale to Output: 48
Grayscale to Output: 83
Grayscale to Output: 105
Grayscale to Output: 66
Grayscale to Output: 63
Grayscale to Output: 102
Grayscale to Output: 108
Grayscale to Output: 110
Grayscale to Output: 121
Grayscale to Output: 176
Grayscale to Output: 191
Grayscale to Output: 132
Grayscale to Output: 114
Grayscale to Output: 128
Grayscale to Output: 97
Grayscale to Output: 119
Grayscale to Output: 144
Grayscale to Output: 135
Grayscale to Output: 112
Grayscale to Output: 121
Grayscale to Output: 132
Grayscale to Output: 138

TIME TAKEN FOR SEQUENTIAL CONVERSION: 0.000728
TIME TAKEN FOR PARALLEL CONVERSION (2 Threads): 0.000372
TIME TAKEN FOR PARALLEL CONVERSION (4 Threads): 0.009483
TIME TAKEN FOR PARALLEL CONVERSION (8 Threads): 0.001894
TIME TAKEN FOR PARALLEL CONVERSION (16 Threads): 0.009239
user@user-VirtualBox:~/Desktop/PC/Lab6$
```



Code:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>
#include <omp.h>

#define IMG_SIZE 300*300

int main()
{
    struct timeval TimeValue_Start;
    struct timezone TimeZone_Start;
    struct timeval TimeValue_Final;
    struct timezone TimeZone_Final;
    long time_start, time_end;
    double time_overhead;

    char filename[100];
    printf("Enter the name of the file containing the pixel values: ");
    scanf("%s", filename);

    FILE* finp = fopen(filename, "r");
    if (!finp)
    {
            printf("Error while opening input file!\n");
            exit(0);
    }

    int R[IMG_SIZE], G[IMG_SIZE], B[IMG_SIZE];
```

```
int a = 0, tmp;
while (!feof(finp) && a < IMG_SIZE)
{
	fscanf(finp, "%d", &tmp);
	R[a] = tmp;

	fscanf(finp, "%d", &tmp);
	G[a] = tmp;

	fscanf(finp, "%d", &tmp);
	B[a] = tmp;

	printf("RGB from Input: %d %d %d\n", R[a], G[a], B[a]);
	a++;
}
fclose(finp);

int GS[IMG_SIZE];

gettimeofday(&TimeValue_Start, &TimeZone_Start);
for (int a = 0; a < IMG_SIZE; a++)
{
	GS[a] = R[a]*0.21+G[a]*0.72+B[a]*0.07;
}
gettimeofday(&TimeValue_Final, &TimeZone_Final);
time_start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
time_end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
time_overhead = (time_end - time_start)/1000000.0;
double SEQ = time_overhead;

gettimeofday(&TimeValue_Start, &TimeZone_Start);
#pragma omp parallel num_threads(2)
{
	#pragma omp for
		for (int a = 0; a < IMG_SIZE; a++)
		{
			GS[a] = R[a]*0.21+G[a]*0.72+B[a]*0.07;
		}
}
gettimeofday(&TimeValue_Final, &TimeZone_Final);
time_start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
time_end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
time_overhead = (time_end - time_start)/1000000.0;
double PAR2 = time_overhead;

gettimeofday(&TimeValue_Start, &TimeZone_Start);
#pragma omp parallel num_threads(4)
{
	#pragma omp for
		for (int a = 0; a < IMG_SIZE; a++)
```

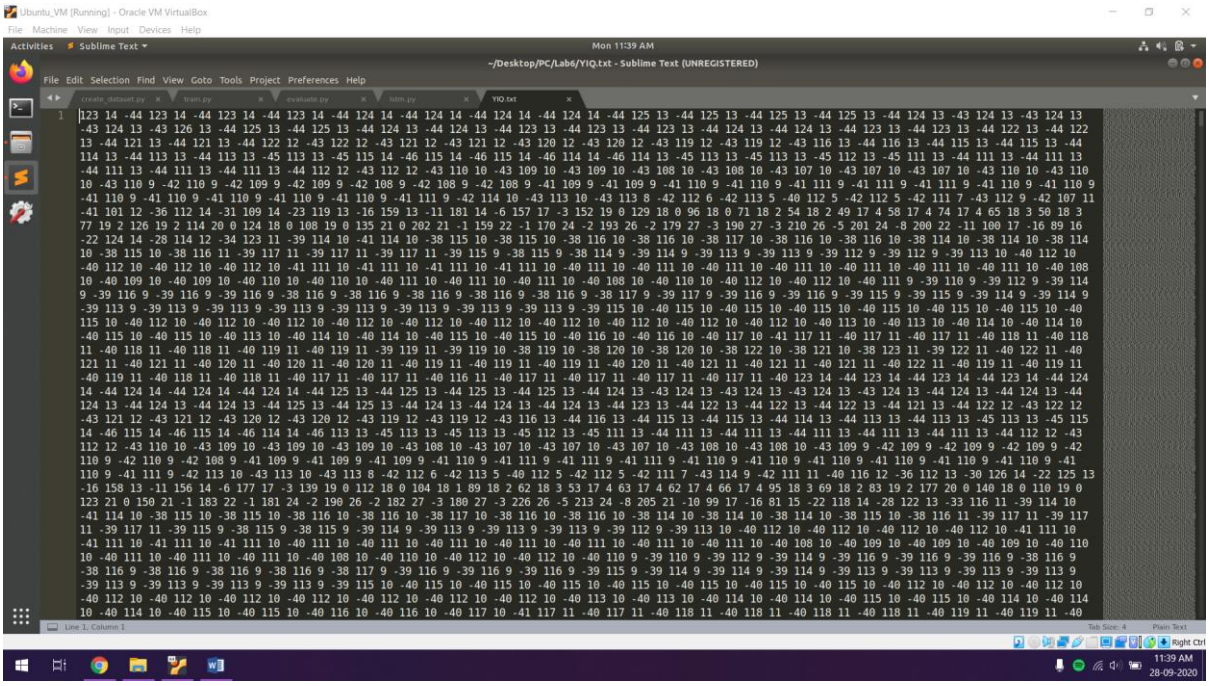
```
                {
                        GS[a] = R[a]*0.21+G[a]*0.72+B[a]*0.07;
                }
}
gettimeofday(&TimeValue_Final, &TimeZone_Final);
time_start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
time_end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
time_overhead = (time_end - time_start)/1000000.0;
double PAR4 = time_overhead;

gettimeofday(&TimeValue_Start, &TimeZone_Start);
#pragma omp parallel num_threads(8)
{
        #pragma omp for
                for (int a = 0; a < IMG_SIZE; a++)
                {
                        GS[a] = R[a]*0.21+G[a]*0.72+B[a]*0.07;
                }
}
gettimeofday(&TimeValue_Final, &TimeZone_Final);
time_start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
time_end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
time_overhead = (time_end - time_start)/1000000.0;
double PAR8 = time_overhead;

gettimeofday(&TimeValue_Start, &TimeZone_Start);
#pragma omp parallel num_threads(16)
{
        #pragma omp for
                for (int a = 0; a < IMG_SIZE; a++)
                {
                        GS[a] = R[a]*0.21+G[a]*0.72+B[a]*0.07;
                }
}
gettimeofday(&TimeValue_Final, &TimeZone_Final);
time_start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
time_end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
time_overhead = (time_end - time_start)/1000000.0;
double PAR16 = time_overhead;

FILE* fout = fopen("Grayscale.txt", "w");
if (!fout)
{
        printf("Error while opening output file!\n");
        exit(0);
}
for (int a = 0; a < IMG_SIZE; a++)
{
        printf("Grayscale to Output: %d\n", GS[a]);
        fprintf(fout, "%d ", GS[a]);
```

```
    }
    fclose(fout);

    printf("\nTIME TAKEN FOR SEQUENTIAL CONVERSION: %lf\n", SEQ);
    printf("TIME TAKEN FOR PARALLEL CONVERSION (2 Threads): %lf\n",
PAR2);
    printf("TIME TAKEN FOR PARALLEL CONVERSION (4 Threads): %lf\n",
PAR4);
    printf("TIME TAKEN FOR PARALLEL CONVERSION (8 Threads): %lf\n",
PAR8);
    printf("TIME TAKEN FOR PARALLEL CONVERSION (16 Threads): %lf\n",
PAR16);
}
```

2. RGB to YIQ:

```
user@user-VirtualBox:~/Desktop/PC/Lab6$ gcc -o rgbToyiq -fopenmp rgbToyiq.c
user@user-VirtualBox:~/Desktop/PC/Lab6$ ./rgbToyiq
Enter the name of the file containing the pixel values: KittenRGB.txt
```

```
File  Edit  View  Search  Terminal  Help
YIQ to Output: 98 9 -42
YIQ to Output: 88 11 -42
YIQ to Output: 86 11 -42
YIQ to Output: 75 9 -37
YIQ to Output: 129 13 -41
YIQ to Output: 95 15 -40
YIQ to Output: 119 15 -40
YIQ to Output: 137 18 -40
YIQ to Output: 74 13 -34
YIQ to Output: 70 12 -32
YIQ to Output: 130 19 -39
YIQ to Output: 139 20 -38
YIQ to Output: 160 19 -39
YIQ to Output: 114 18 -40
YIQ to Output: 91 16 -41
YIQ to Output: 157 14 -38
YIQ to Output: 185 11 -39
YIQ to Output: 88 8 -37
YIQ to Output: 77 5 -37
YIQ to Output: 42 -5 -28
YIQ to Output: 76 1 -35
YIQ to Output: 98 1 -34
YIQ to Output: 59 2 -33
YIQ to Output: 56 2 -32
YIQ to Output: 95 8 -36
YIQ to Output: 101 12 -35
YIQ to Output: 103 16 -36
YIQ to Output: 114 17 -37
YIQ to Output: 169 17 -38
YIQ to Output: 183 11 -40
YIQ to Output: 124 9 -42
YIQ to Output: 106 12 -41
YIQ to Output: 120 14 -42
YIQ to Output: 89 15 -42
YIQ to Output: 111 17 -43
YIQ to Output: 136 20 -42
YIQ to Output: 127 21 -44
YIQ to Output: 105 23 -43
YIQ to Output: 113 25 -45
YIQ to Output: 125 25 -44
YIQ to Output: 130 26 -45

TIME TAKEN FOR SEQUENTIAL CONVERSION: 0.001855
TIME TAKEN FOR PARALLEL CONVERSION (2 Threads): 0.000930
TIME TAKEN FOR PARALLEL CONVERSION (4 Threads): 0.004001
TIME TAKEN FOR PARALLEL CONVERSION (8 Threads): 0.003236
TIME TAKEN FOR PARALLEL CONVERSION (16 Threads): 0.001381
user@user-VirtualBox:~/Desktop/PC/Lab6$
```



Code:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>
#include <omp.h>

#define IMG_SIZE 300*300

int main()
{
    struct timeval TimeValue_Start;
    struct timezone TimeZone_Start;
    struct timeval TimeValue_Final;
    struct timezone TimeZone_Final;
    long time_start, time_end;
    double time_overhead;

    char filename[100];
    printf("Enter the name of the file containing the pixel values: ");
    scanf("%s", filename);

    FILE* finp = fopen(filename, "r");
    if (!finp)
    {
            printf("Error while opening input file!\n");
            exit(0);
    }

    int R[IMG_SIZE], G[IMG_SIZE], B[IMG_SIZE];
```

```
int a = 0, tmp;
while (!feof(finp) && a < IMG_SIZE)
{
	fscanf(finp, "%d", &tmp);
	R[a] = tmp;

	fscanf(finp, "%d", &tmp);
	G[a] = tmp;

	fscanf(finp, "%d", &tmp);
	B[a] = tmp;

	printf("RGB from Input: %d %d %d\n", R[a], G[a], B[a]);
	a++;
}
fclose(finp);

int Y[IMG_SIZE], I[IMG_SIZE], Q[IMG_SIZE];

gettimeofday(&TimeValue_Start, &TimeZone_Start);
for (int a = 0; a < IMG_SIZE; a++)
{
	Y[a] = 0.299*R[a] + 0.587*G[a] + 0.114*B[a];
	I[a] = 0.596*R[a] - 0.275*G[a] - 0.321*B[a];
	Q[a] = 0.212*R[a] - 0.523*G[a] + 0.311*B[a];

}
gettimeofday(&TimeValue_Final, &TimeZone_Final);
time_start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
time_end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
time_overhead = (time_end - time_start)/1000000.0;
double SEQ = time_overhead;

gettimeofday(&TimeValue_Start, &TimeZone_Start);
#pragma omp parallel num_threads(2)
{
	#pragma omp for
		for (int a = 0; a < IMG_SIZE; a++)
		{
			Y[a] = 0.299*R[a] + 0.587*G[a] + 0.114*B[a];
			I[a] = 0.596*R[a] - 0.275*G[a] - 0.321*B[a];
			Q[a] = 0.212*R[a] - 0.523*G[a] + 0.311*B[a];
		}
}
gettimeofday(&TimeValue_Final, &TimeZone_Final);
time_start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
time_end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
time_overhead = (time_end - time_start)/1000000.0;
double PAR2 = time_overhead;
```

```
gettimeofday(&TimeValue_Start, &TimeZone_Start);
#pragma omp parallel num_threads(4)
{
	#pragma omp for
		for (int a = 0; a < IMG_SIZE; a++)
		{
			Y[a] = 0.299*R[a] + 0.587*G[a] + 0.114*B[a];
			I[a] = 0.596*R[a] - 0.275*G[a] - 0.321*B[a];
			Q[a] = 0.212*R[a] - 0.523*G[a] + 0.311*B[a];
		}
}
gettimeofday(&TimeValue_Final, &TimeZone_Final);
time_start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
time_end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
time_overhead = (time_end - time_start)/1000000.0;
double PAR4 = time_overhead;

gettimeofday(&TimeValue_Start, &TimeZone_Start);
#pragma omp parallel num_threads(8)
{
	#pragma omp for
		for (int a = 0; a < IMG_SIZE; a++)
		{
			Y[a] = 0.299*R[a] + 0.587*G[a] + 0.114*B[a];
			I[a] = 0.596*R[a] - 0.275*G[a] - 0.321*B[a];
			Q[a] = 0.212*R[a] - 0.523*G[a] + 0.311*B[a];
		}
}
gettimeofday(&TimeValue_Final, &TimeZone_Final);
time_start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
time_end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
time_overhead = (time_end - time_start)/1000000.0;
double PAR8 = time_overhead;

gettimeofday(&TimeValue_Start, &TimeZone_Start);
#pragma omp parallel num_threads(16)
{
	#pragma omp for
		for (int a = 0; a < IMG_SIZE; a++)
		{
			Y[a] = 0.299*R[a] + 0.587*G[a] + 0.114*B[a];
			I[a] = 0.596*R[a] - 0.275*G[a] - 0.321*B[a];
			Q[a] = 0.212*R[a] - 0.523*G[a] + 0.311*B[a];
		}
}
gettimeofday(&TimeValue_Final, &TimeZone_Final);
time_start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
time_end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
time_overhead = (time_end - time_start)/1000000.0;
double PAR16 = time_overhead;
```

```
    FILE* fout = fopen("YIQ.txt", "w");
    if (!fout)
    {
            printf("Error while opening output file!\n");
            exit(0);
    }
    for (int a = 0; a < IMG_SIZE; a++)
    {
            printf("YIQ to Output: %d %d %d\n", Y[a], I[a], Q[a]);
            fprintf(fout, "%d %d %d ", Y[a], I[a], Q[a]);
    }
    fclose(fout);

    printf("\nTIME TAKEN FOR SEQUENTIAL CONVERSION: %lf\n", SEQ);
    printf("TIME TAKEN FOR PARALLEL CONVERSION (2 Threads): %lf\n",
PAR2);
    printf("TIME TAKEN FOR PARALLEL CONVERSION (4 Threads): %lf\n",
PAR4);
    printf("TIME TAKEN FOR PARALLEL CONVERSION (8 Threads): %lf\n",
PAR8);
    printf("TIME TAKEN FOR PARALLEL CONVERSION (16 Threads): %lf\n",
PAR16);
}
```