Implementation of image processing techniques using openmp and mpi

Sharuf Baig(181EC143)
Department of Electronics and communication engineering
National institute of technology Karnataka
Surathkal,Karnataka,India
sharufbaig321@gmail.com

Surya Prakash Reddy(181EC234)
Department of Electronics and communication engineering
National institute of technology Karnataka
Surathkal,Karnataka,India
putlurusurya2000@gmail.com

Phanindra Sai Kumar(181EC230)
Department of Electronics and communication engineering
National institute of technology Karnataka
Surathkal,Karnataka,India
phanindrasai21@gmail.com

***Abstract*—Digital Image processing is very useful in many applications such as remote sensing, medical, vision applications etc. There are a large number of techniques in Digital image processing. These techniques involve a lot of computations and hence require a large amount of time to complete if done sequentially. The time required can be reduced by parallelising it. In this project we implement some of the image processing techniques using openmp.**

Keywords—openmp, image processing, convolution, contrast, edge detection, blurring

I. Introduction

With advancement in image capturing machines and other technology, image size has increased from past size. For big size data, processing time is longer. Image processing includes image recognition, image detection, face detection, etc. Especially in image detection time consumption is high. For image detection, various techniques are used like SVM, HOG, BRIEF, etc. It can be reduced with help of parallelization. In general terms, the parallelism means computation done by two or more computing units simultaneously.With the help of this library the processing time of image processing tasks can be reduced drastically.

II. SIMILAR WORK

There are many research papers about OpenMP and different techniques of image processing. Object detection and recognition, image classification, image feature extraction are works, which are done in an image processing. For these things, lots of time is required. Hence, we can use OpenMP which can be useful for reducing time consumption. With multicore and many core processors, OpenMP gives notable performance.

Image convolution is widely used for sharpening, blurring, and edge detection. In [2], they reviewed two common algorithms for convolving a 2D image by a separable kernel (filter). After optimizing the sequential codes using loop unrolling and SIMD vectorization, they chose the algorithm with better performance as the baseline for parallelization. They then compare the parallel performance of the optimized code using OpenMP, OpenCL, and GPRM implementations on the Intel Xeon Phi. In[3], a comparative analysis of the execution speed of a multi-core platform which is involved in the parallel execution of Discrete Fourier Transform (DFT) and Inverse Discrete Fourier Transform (IDFT) programs over sequential ones is done

II. METHODOLOGY

Image processing means performing some operations on input data such as image (including photograph or video frame) for getting information and clear image as output. Usually in Image Processing system, when applying already set signal processing methods to images, they are deeming as 2D signals

In this project we aim to improve computation time of the following techniques using openmp parallelisation techniques.

A. 2D Convolution

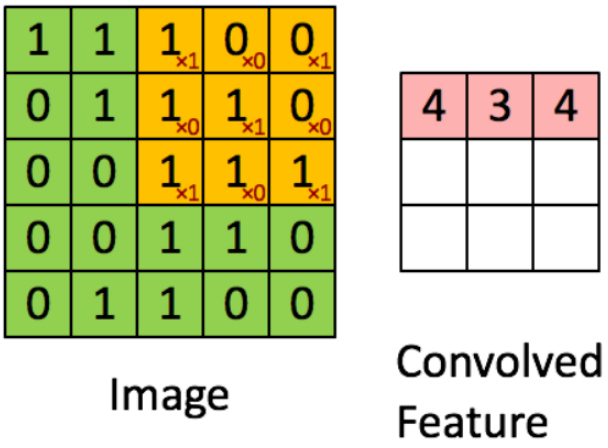

Figure 1:- Convolution

2D convolution is the main machinery behind most forms of image processing. From computer vision to feature detection, from processing MRI scans to making a selfie look like pencil sketches, 2D convolution is the main computation that is involved in a large portion of image processing algorithms. So it is greatly desired to have a very fast implementation of this computationally intensive operation, since image processing relies heavily upon it.Essentially, 2D convolution is taking a small matrix, called the kernel, and sweeping it across an

image, which is a 2D matrix of values that correspond to what is in each pixel.

```
for(y = kCenterX; y < rows-kCenterX; y++)
for( x = 0; x <= cols-1; x++)
for( k=0;k<krows;k++)
for( j=0;j<kcols;j++){

   q = y-kCenterX ;
   p = x-kCenterY ;
   if(q+k>=0 && q+k<rows && p+j>=0 && p+j<cols){
       dst[y-kCenterX][x]+=src[q+k][p+j]*h[k][j];
   }

}
```

Sequential convolution pseudo code

B. *Brightness*

Brightness refers to the overall lightness or darkness of the image.A color screen uses three color schemes i.e., RGB scheme (red, green and blue).The brightness of the screen depends upon the sum of the amplitude of red green and blue pixels, and it is divided by 3.

$$I = \frac{1}{3}(R + G + B)$$

where I is the brightness of the pixel and R,G,B are respective red,green and blue values of the respective pixel.

To increase the brightness we need to increase the intensity of each pixel by a constant and similarly to darken the image we need to decrease the intensity of every pixel of the image.Truncation is used to keep the value in the valid range i.e. 0 to 255. If the value goes below 0 it will get truncated to zero and if the value goes above 255 it will get truncated to 255.When the brightness is decreased, the color appears dull,and when brightness increases, the color is clearer.

C. *Contrast*

The term contrast refers to the amount of color or grayscale differentiation that exists between various image features in digital images.In simple terms,it can be explained as the difference between maximum and minimum pixel intensity in an image. Images having a higher contrast level generally display a greater degree of color or grayscale variation than those of lower contrast.

Truncation is used to keep the value in the valid range i.e. 0 to +255. The procedure for it is same as that as in Section B of methodology.The value of contrast will be in the range of -255 to +255. Negative values will decrease the amount of contrast and, conversely, positive values will increase the amount of contrast.

D. *Image Rotation*

Image rotation is a common image processing routine with applications in matching, alignment, and other image-based algorithms. The input to an image rotation routine is an image, the rotation angle θ, and a point about which rotation is done.

The coordinates of a point (x1, y1) when rotated by an angle θ around (x0, y0) become (x2, y2), as shown by the following equation:

$$x_2 = \cos(\theta) * (x_1 - x_0) + \sin(\theta) * (y_1 - y_0)$$
$$y_2 = -\sin(\theta) * (x_1 - x_0) + \cos(\theta) * (y_1 - y_0)$$

We can see that the calculations of the new (x, y) coordinate of each pixel in the input can be done independently.Hence it can be implemented in a parallel manner.

E. *Color Spaces Conversion*

Color space conversion is the translation of the representation of a color from one basis to another. This typically occurs in the context of converting an image that is represented in one color space to another color space, the goal being to make the translated image look as similar as possible to the original.

F. Blur

An image contains a lot of features like edge, contrast etc. In image processing features have to be extracted from the image for further study of image. Blurring is one of the image processing techniques which makes the image smooth. A filter used for blurring is also called a low pass filter, because it allows low frequency to enter and stop high frequency. Here frequency means the change of pixel value. Around the edge, pixel value changes rapidly as the blurred image is smooth so high frequency should be filtered out. For blur purposes, a filter with every call having value 1 is used because to blur an image a pixel value should be close to the neighbor value. The filter is divided by the size of the filter for normalization.

```
int m,n,ii=0,jj=0;
for(i=0; i < rows; ++i)
{

    for(j=0; j < cols; ++j)
    {
        int l = min(i+kRows,rows),k=min(j+kCols,cols);
        double sum=0;
        #pragma omp parallel num_threads(no_threads) shared(sum)
        {
            #pragma omp for private(m,n) collapse(2) reduction(+:sum)
            for(m=i; m < l;m++)
            {

                for(n=j; n <k;n++)
                {
                    sum+=img1.arr[m][n]*f.arr[m-i][n-j];
                }
            }
        }
        if(i<=rows-kRows and j<=cols-kCols){
            out.arr[ii][jj]+=sum;
            jj++;
        }
    }
    if(i<=rows-kRows)ii++;
    jj=0;
}
```

Caring out Blurring using a gaussian filter

G. *Edge Detection*

An edge can be defined as a set of connected pixels that forms a boundary between two disjoint regions.Edge Detection is a method of segmenting an image into regions of discontinuity. It is a widely used technique in digital image processing for applications like pattern recognition,image morphology and feature extraction.

Edge detection allows users to observe the features of and image for a significant change in the gray level. This texture indicates the end of one region in the image and the beginning of another. It reduces the amount of data in an image and preserves the structural properties of an image.

IV. RESULTS

A.Convolution

The 2D convolution function written was tested with Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz with 4 cores and 8 threads. The input of the function is a 2D matrix of size n*n. The function is tested with 3*3 filter.

```
fidi@fidigaadu:~$ ./out1
time   = 0.000056 secs
        34      70      109     148     187     226     265     304     343     164
        70      133     199     265     331     397     463     529     595     280
        109     199     292     385     478     571     664     757     850     397
        148     265     385     505     625     745     865     985     1105    514
        187     331     478     625     772     919     1066    1213    1360    631
        226     397     571     745     919     1093    1267    1441    1615    748
        265     463     664     865     1066    1267    1468    1669    1870    865
        304     529     757     985     1213    1441    1669    1897    2125    982
        343     595     850     1105    1360    1615    1870    2125    2380    1099
        164     280     397     514     631     748     865     982     1099    475
```

Convulution output of 10*10 image with 3*3 filter

Performance of different models of the 2D convolution is shown in the below table. All models are tested with the same input array and same filter. number of threads used for openmp model is 8. number of processes used for mpi model is 4.

size-> type	100	500	1000	2500	5000	10000	15000
sequential	0.003044 secs	0.032586 secs	0.077949 secs	0.394676 secs	1.572499 secs	6.297465 secs	14.151098 secs
openmp	0.015190 secs	0.018623 secs	0.028652 secs	0.144873 secs	0.564325 secs	2.278071 secs	5.120126 secs
hybrid(openmp+mpi)	0.095034 secs	0.105973 secs	0.112780 secs	0.368725 secs	1.253264 secs	5.484540 secs.	12.256402 secs

Table: Performance of different models

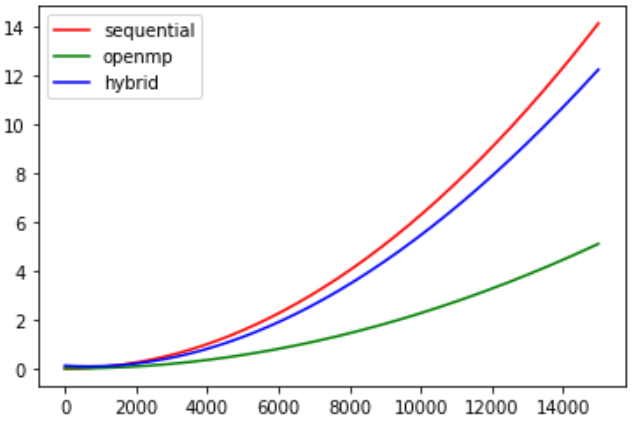

Fig: Graph of the Performance of different models

B.EDGE DETECTION

Edge detection is achieved by using filter of users choice like sobel,laplacian etc. Performance of this is similar to that if 2D convolution. In for performance evaluation vertical edge detection filter was used.

```
fidi@fidigaadu:~$ ./out1
time   = 0.000066 secs
        0       0       0       0       0       0       0       0       0       0
        -1      -3      -6      -9      -12     -15     -18     -21     -24     -17
        -4      -9      -15     -21     -27     -33     -39     -45     -51     -36
        -7      -15     -24     -33     -42     -51     -60     -69     -78     -55
        -10     -21     -33     -45     -57     -69     -81     -93     -105    -74
        -13     -27     -42     -57     -72     -87     -102    -117    -132    -93
        -16     -33     -51     -69     -87     -105    -123    -141    -159    -112
        -19     -39     -60     -81     -102    -123    -144    -165    -186    -131
        -22     -45     -69     -93     -117    -141    -165    -189    -213    -150
        -25     -51     -78     -105    -132    -159    -186    -213    -240    -169
fidi@fidigaadu:~$
```

Fig: Edge detection output

C. Blur

Blurring is performed by convoluting the image with a gaussian filter of preferred size and is tested on Intel® Core™ i5-8250U CPU @ 1.60GHz × 8 with 4 threads and the output obtained is as follows.

Fig. Original image

Fig. Blurred image using a gaussian filter of size 5*5

D. Rotation

The image is rotated here by the specified angle and is tested on similar conditions and the output obtained is as follows

Fig: Rotation of image by 90 degree

E. Color Space conversion

The image is converted from one color space to another like RGB to YUV and YUV to RGB and RGB to grayscale and the output obtained is as follows

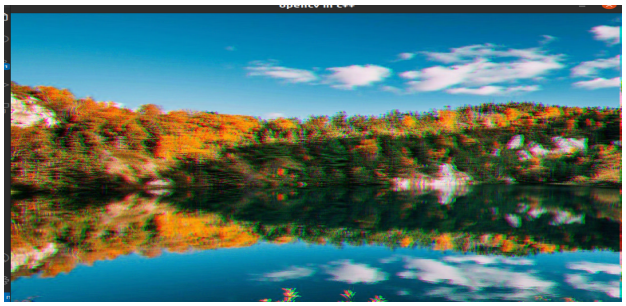

Fig:- RGB input image

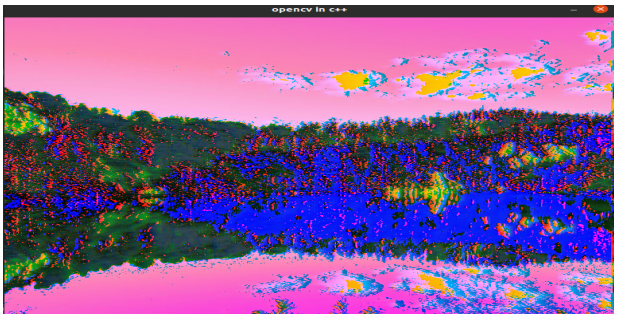

Fig:- YUV output image

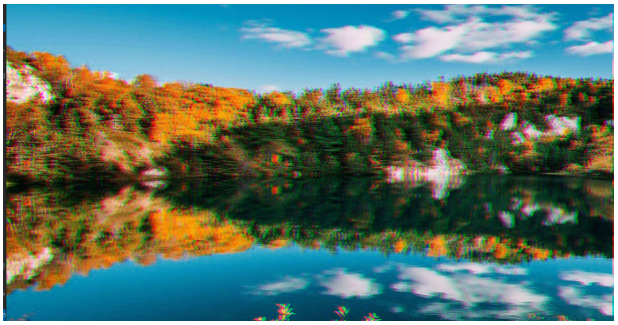

Fig:- Obtained RGB image from YUV

Fig: Obtained Grayscale image from RGB image

F. BRIGHTNESS

The brightness of the image is increased and decreased to the original respectively. The results are obtained as follows.

Fig :- Decreased brightness image

Fig :- Increased brightness image

G. CONTRAST

The contrast of the image is increased and decreased to the original respectively. The results are obtained as follows.

Fig :- Decreased contrast image

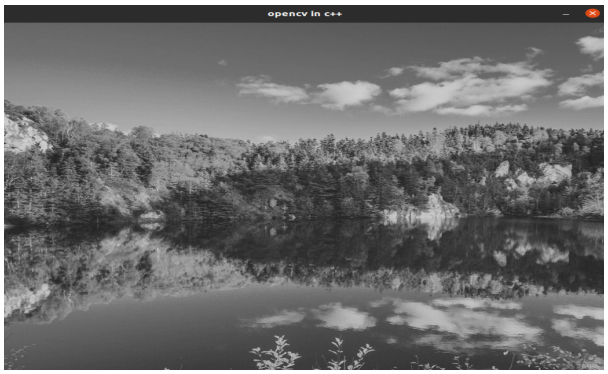

Fig :- Increased contrast image

OPERATION	768*768 IMAGE		1024*1024 IMAGE		1600*1200 IMAGE	
	SERIAL	PARALLEL	SERIAL	PARALLEL	SERIAL	PARALLEL
BLUR	5.160635	4.029612	9.133219	6.828968	16.205204	11.952847
ROTATION	0.008867	0.005662	0.010195	0.010246	0.031327	0.008120
RGB TO YUV PRINTING	0.145396	0.111477	0.222832	0.166366	0.446859	0.252506
RGB TO YUV CONVERSION	0.048883	0.021465	0.094435	0.036669	0.165735	0.061237
YUV TO RGB PRINTING	0.048480	0.023778	0.102521	0.036589	0.172769	0.057151
YUV TO RGB CONVERSION	0.108498	0.108250	0.150995	0.128366	0.284436	0.229247
RGB TO GRAY PRINTING	0.054774	0.020515	0.097154	0.037021	0.179198	0.071449
RGB TO GRAY CONVERSION	0.041848	0.040938	0.050278	0.053958	0.095885	0.101942
CONTRAST	0.014434	0.007550	0.030654	0.014213	0.045866	0.013239
BRIGHTNESS	0.019539	0.004992	0.026743	0.012673	0.090506	0.011592

Table:- Time in milliseconds to perform each operation serial and parallel fashion taken for different sizes of images

V. CONTRIBUTIONS

Surya Prakash:-
Convolution, Edge detection

Sharuf Baig:-
Blur, Image Rotation, Color spaces conversion

Phanindra:-
Brightness, Contrast

VII. FURTHER IMPLEMENTATIONS

These algorithms can be further improved using Cuda programming. The hybrid model is tested using only 4 processes due to resource constraints. The hybrid model at lower sizes gives a promising performance. The hybrid model achieves only single dimensional parallelism with mpi I.e entire rows are shared among the processes. This can be improved to divide both rows and columns among the processes.

VIII. CONCLUSION

Important image processing techniques like 2D convolution, edge detection, changing color spaces, brightness contrast changing etc are implemented using parallel computing techniques. Convolution 2D is implemented using sequential openmp and hybrid(openmp+mpi) models. All the models tested with Intel i5 8th gen cpu. Parallel processing models showed reduction of almost 50% of the processing time compared to sequential models.

IX. REFERENCES

[1] Ashkan Tousimojarad, Wim Vanderbauwhede, W Paul Cockshott, “2D Image Convolution using Three Parallel Programming Models on the Xeon Phi”

[2] Sen Pan,Russ Miller”Parallel Image Blurring With OpenMP”. (references)

[3] Prof. Kotresh Marali, Irshadahmed Preerjade, Shreevatsa Tilgul, Shrikrishna K, Sourab M Kulkarni, 0, Performance Analysis of Fourier Transform Algorithms with and without using OpenMP,