PROJECT 1 OPENMP

NAMES: NTONES SAVVAS, PETROUDIS NIKOLAOS
PROFESSOR: IRAKLIS SPILIOTIS
SUBJECT: PARALLEL PROCESSING TECHNOLOGY
TOPIC: BINARIZATION OF GREY IMAGES USING OTSU METHOD
LABORATORY OF COMPUTER ARCHTECTURE AND HIGH-PERFORMANCE SYSTEMS
DEPARTURE OF ELECTRICAL AND COMPUTER ENGINEERING
DEMOCRETUS UNIVERSITY OF THRACE
ACADEMIC SEASON: 2022 – 2023

ΔΗΜΟΚΡΙΤΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΡΑΚΗΣ

ΤΜΗΜΑ
ΗΜ & ΜΥ

INTRODUCTION

A binary image is created by dividing a gray-scale image into two levels, usually called area 0 and area 1 [2]. Thresholding is an important technique of computer vision and image processing, by which a target-object is separated from the background image [1]. Using Otsu method, a threshold is found automatically in order image binarization to be achieved [2]. The optimal threshold is chosen by the classification criterion to separate the two categories (foreground and background. Specifically, threshold is determined by minimizing intra-class intensity or by maximizing inter-class variance, which is defined as a weighted variance of average intensity values of the two categories. [1].

As a result, image is separated into two areas, the bright area T0 and the dark area T1. T0 area contains intensity levels from 0 to t, while T1 region contains intensity levels from t to l, where t is the threshold and l is the maximum intensity level (i.e. 256). T0 and T1 can correspond to object and background or vice versa (the bright area does not always correspond to the object) [4].

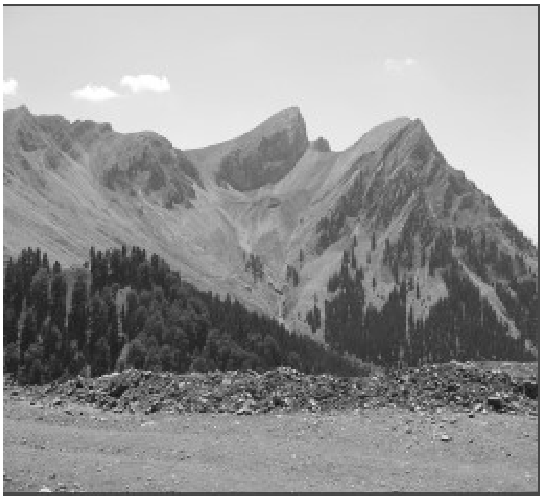


IMAGE 1 – Mountain 1024x1024

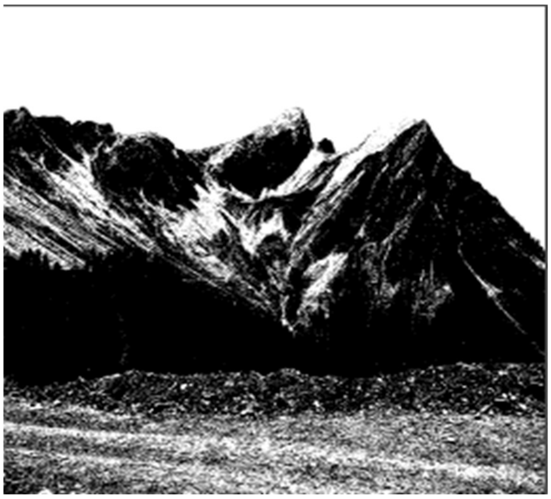


IMAGE 2 – Mountain 1024x1024 IMAGE WITH OTSU

PSEUDOCODE

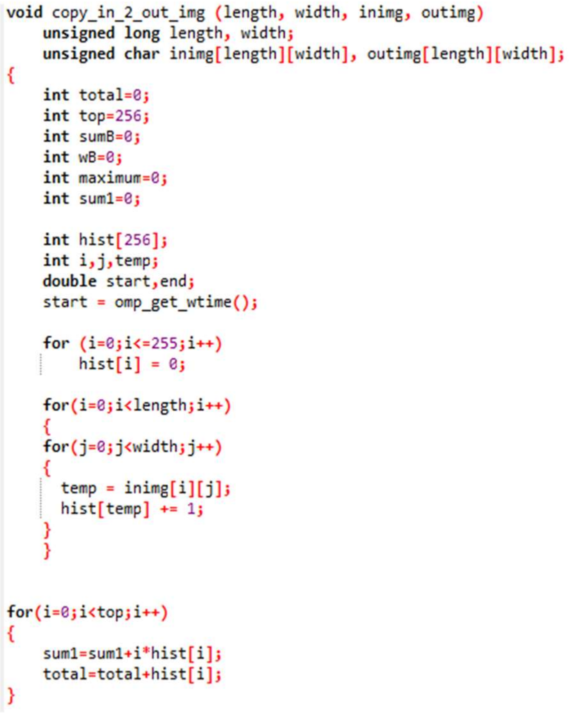
STEP 1: Histogram computation of grey-scale image.

STEP 2 (Repeated) : Foreground and background variance computation for one threshold value each time.

```
    i)      Computation of foreground and background pixels’ weight.
    ii)     Computation of foreground and background pixels’ average value.
    iii)    Computation of foreground and background pixels’ variance.
```

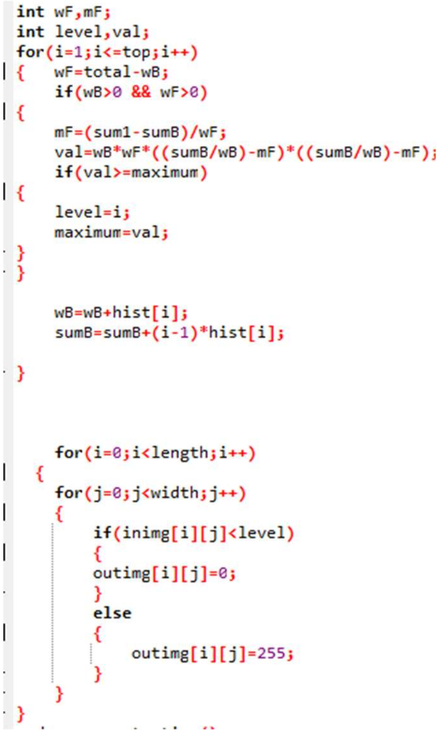
STEP 3: Inter-class variance computation and the optimal threshold is determined by the maximum value of inter-class variance.

STEP 4: Intensity values less than the threshold value are determined to new intensity value 0, otherwise 255.



```
void copy_in_2_out_img (length, width, inimg, outimg)
    unsigned long length, width;
    unsigned char inimg[length][width], outimg[length][width];
{
    int total=0;
    int top=256;
    int sumB=0;
    int wB=0;
    int maximum=0;
    int sum1=0;

    int hist[256];
    int i,j,temp;
    double start,end;
    start = omp_get_wtime();

    for (i=0;i<=255;i++)
        hist[i] = 0;

    for(i=0;i<length;i++)
    {
    for(j=0;j<width;j++)
    {
      temp = inimg[i][j];
      hist[temp] += 1;
    }
    }


for(i=0;i<top;i++)
{
    sum1=sum1+i*hist[i];
    total=total+hist[i];
}
```

Image 3 – Serial code 1

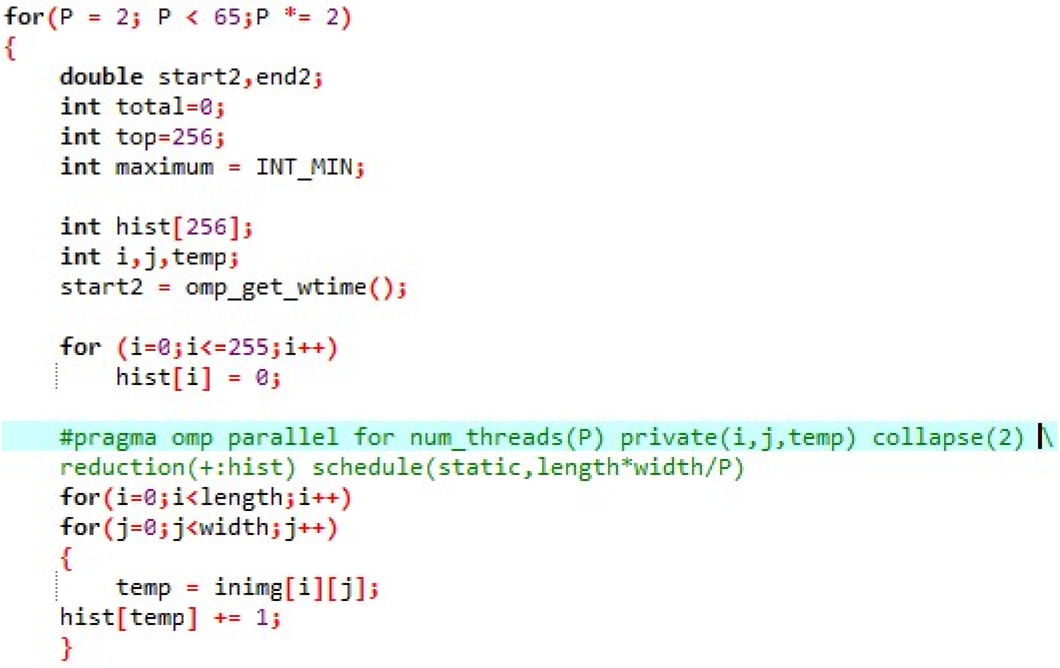


```
int wF,mF;
int level,val;
for(i=1;i<=top;i++)
{   wF=total-wB;
    if(wB>0 && wF>0)
{
    mF=(sum1-sumB)/wF;
    val=wB*wF*((sumB/wB)-mF)*((sumB/wB)-mF);
    if(val>=maximum)
{
    level=i;
    maximum=val;
}
}


    wB=wB+hist[i];
    sumB=sumB+(i-1)*hist[i];

}



    for(i=0;i<length;i++)
  {
    for(j=0;j<width;j++)
    {
        if(inimg[i][j]<level)
        {
        outimg[i][j]=0;
        }
        else
        {
            outimg[i][j]=255;
        }
    }
}
```

Image 4 – Serial code 2

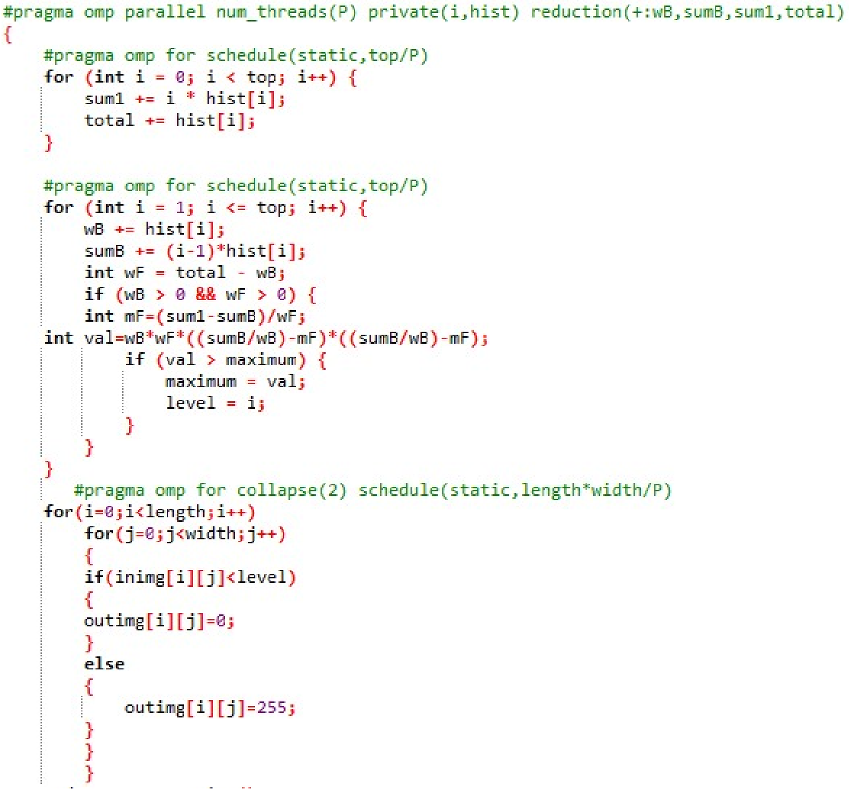
The time complexity of this algorithm is O(N*M + 2*K), where N,M are the height and width of image respectively and K is the maximum intensity value of grey-scale image.

Code’s parallelization



```
for(P = 2; P < 65;P *= 2)
{
    double start2,end2;
    int total=0;
    int top=256;
    int maximum = INT_MIN;

    int hist[256];
    int i,j,temp;
    start2 = omp_get_wtime();

    for (i=0;i<=255;i++)
        hist[i] = 0;

    #pragma omp parallel for num_threads(P) private(i,j,temp) collapse(2) \
    reduction(+:hist) schedule(static,length*width/P)
    for(i=0;i<length;i++)
    for(j=0;j<width;j++)
    {
        temp = inimg[i][j];
    hist[temp] += 1;
    }
```

Image 5 – Parallel code 1



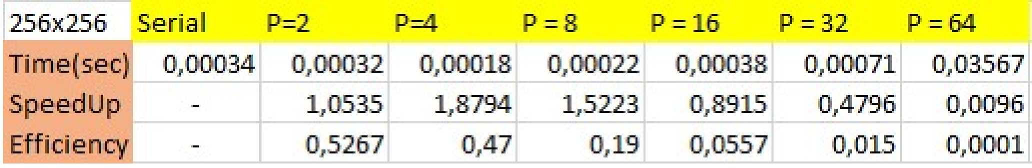
```
#pragma omp parallel num_threads(P) private(i,hist) reduction(+:wB,sumB,sum1,total)
{
    #pragma omp for schedule(static,top/P)
    for (int i = 0; i < top; i++) {
        sum1 += i * hist[i];
        total += hist[i];
    }

    #pragma omp for schedule(static,top/P)
    for (int i = 1; i <= top; i++) {
        wB += hist[i];
        sumB += (i-1)*hist[i];
        int wF = total - wB;
        if (wB > 0 && wF > 0) {
        int mF=(sum1-sumB)/wF;
    int val=wB*wF*((sumB/wB)-mF)*((sumB/wB)-mF);
            if (val > maximum) {
                maximum = val;
                level = i;
            }
        }
    }
       #pragma omp for collapse(2) schedule(static,length*width/P)
    for(i=0;i<length;i++)
        for(j=0;j<width;j++)
        {
        if(inimg[i][j]<level)
        {
        outimg[i][j]=0;
        }
        else
        {
            outimg[i][j]=255;
        }
        }
        }
```

Image 6 -Parallel code 2

Metrics

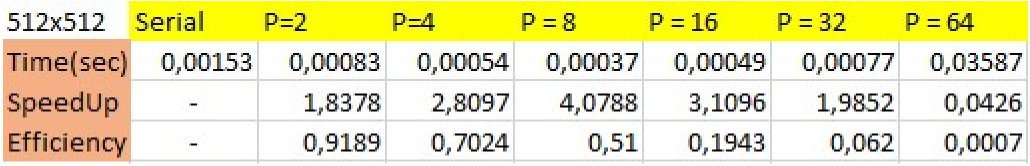
Metrics for different image sizes are presented below. These were calculated in pxeon2 a machine that includes 64 processors and has 128GB RAM.

IMAGE 256X256 METRICS



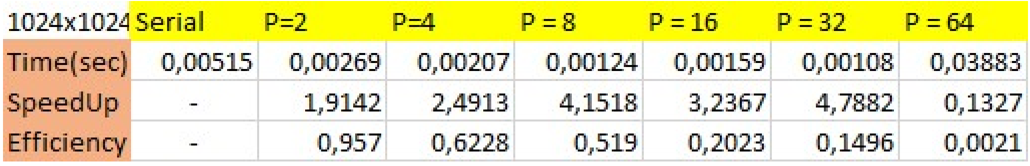
256x256	Serial	P=2	P=4	P = 8	P = 16	P = 32	P = 64
Time(sec)	0,00034	0,00032	0,00018	0,00022	0,00038	0,00071	0,03567
SpeedUp	-	1,0535	1,8794	1,5223	0,8915	0,4796	0,0096
Efficiency	-	0,5267	0,47	0,19	0,0557	0,015	0,0001

IMAGE 512x512 METRICS



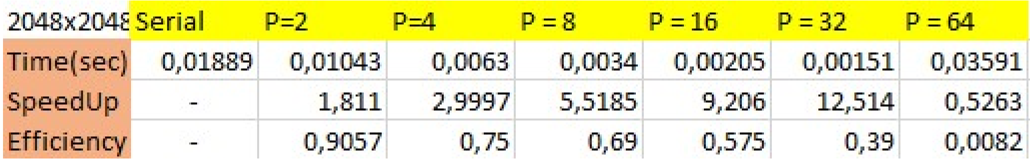
512x512	Serial	P=2	P=4	P = 8	P = 16	P = 32	P = 64
Time(sec)	0,00153	0,00083	0,00054	0,00037	0,00049	0,00077	0,03587
SpeedUp	-	1,8378	2,8097	4,0788	3,1096	1,9852	0,0426
Efficiency	-	0,9189	0,7024	0,51	0,1943	0,062	0,0007

IMAGE 1024X1024 METRICS



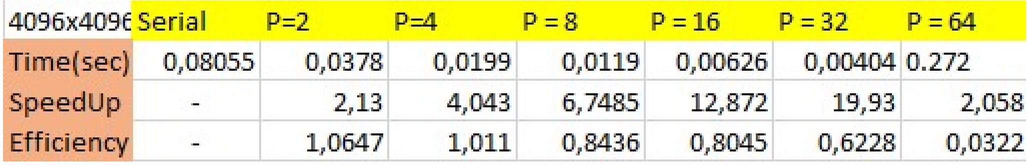
1024x1024	Serial	P=2	P=4	P = 8	P = 16	P = 32	P = 64
Time(sec)	0,00515	0,00269	0,00207	0,00124	0,00159	0,00108	0,03883
SpeedUp	-	1,9142	2,4913	4,1518	3,2367	4,7882	0,1327
Efficiency	-	0,957	0,6228	0,519	0,2023	0,1496	0,0021

IMAGE 2048X2048 METRICS



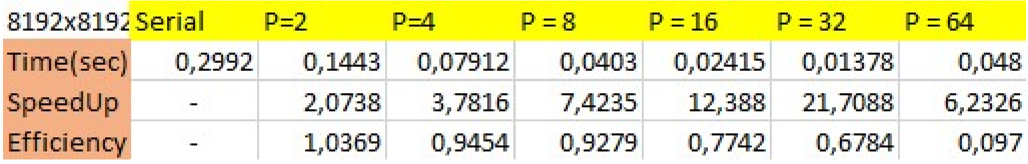
2048x2048	Serial	P=2	P=4	P = 8	P = 16	P = 32	P = 64
Time(sec)	0,01889	0,01043	0,0063	0,0034	0,00205	0,00151	0,03591
SpeedUp	-	1,811	2,9997	5,5185	9,206	12,514	0,5263
Efficiency	-	0,9057	0,75	0,69	0,575	0,39	0,0082

IMAGE 4096x4096 METRICS



4096x4096	Serial	P=2	P=4	P = 8	P = 16	P = 32	P = 64
Time(sec)	0,08055	0,0378	0,0199	0,0119	0,00626	0,00404	0.272
SpeedUp	-	2,13	4,043	6,7485	12,872	19,93	2,058
Efficiency	-	1,0647	1,011	0,8436	0,8045	0,6228	0,0322

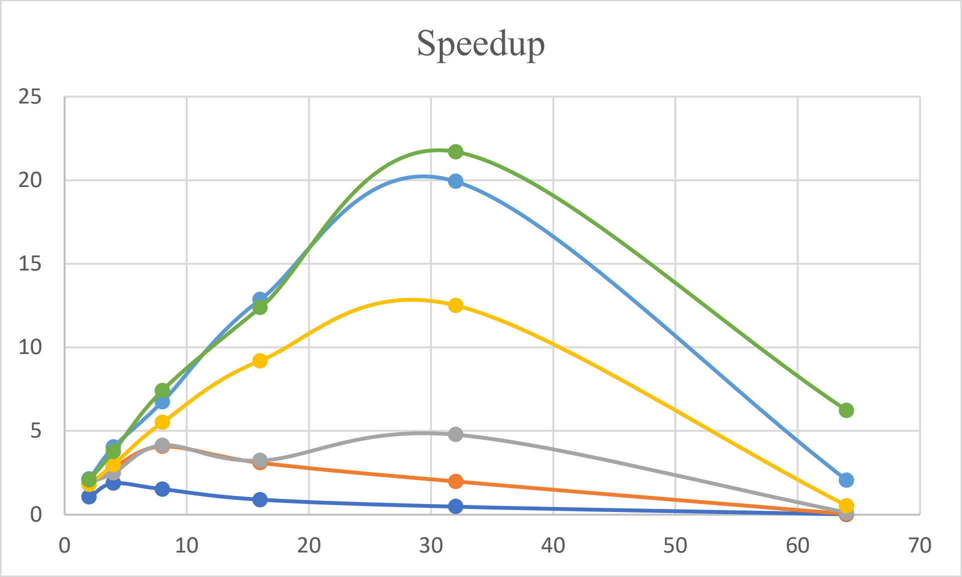
IMAGE 8192x8192 METRICS



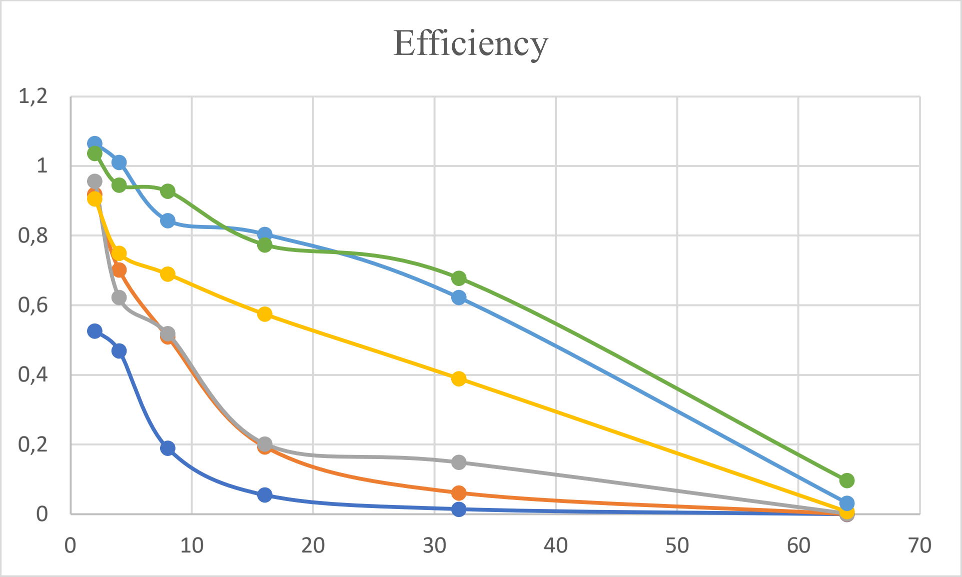
8192x8192	Serial	P=2	P=4	P = 8	P = 16	P = 32	P = 64
Time(sec)	0,2992	0,1443	0,07912	0,0403	0,02415	0,01378	0,048
SpeedUp	-	2,0738	3,7816	7,4235	12,388	21,7088	6,2326
Efficiency	-	1,0369	0,9454	0,9279	0,7742	0,6784	0,097

GRAPHS

Speedup and efficiency graphs are presented below.



GRAPH 1 - SPEEDUP – PROCESSORS



GRAPH 2 - EFFICIENCY - CORES

CONCLUSION

Code created to perform image processing with the Otsu method is parallelizable up to 32 cores for images with a resolution of 2048x2048 and above. In terms of scalability, the generated parallel code is scalable for images with a resolution greater than 1024x1024 since the speedup increases by adding more processors to the parallelization [3]

Bibliography

1)Nobuyuki Otsu (1979). "A threshold selection method from gray-level histograms". *IEEE Trans. Sys. Man. Cyber*. **9** (1): 62–66. https://ieeexplore.ieee.org/document/4310076

2) Sunil L. Bangare, Amruta Dubal, Pallavi S. Bangare, Dr. S.T. Patil, Reviewing Otsu's Method For Image Thresholding, International Journal of Applied Engineering Research, https://dx.doi.org/10.37622/IJAER/10.9.2015.21777-21783

3) An introduction to parallel programming, Peter S. Pacheco

4) Jamileh Yousefi, Image Binarization using Otsu Thresholding Algorithm, http://dx.doi.org/10.13140/RG.2.1.4758.9284