IMAGE DITHERING

A look at parallelization

Anurag Malyala

2K15/CO/035

PA- Slot (D)

Note: This ppt is part 4/4 of a series of ppts by : {/CO/125, /CO/015, /CO/119}

INTRODUCTION

RECAP
DITHERING (DEF)

- Dithering refers to creating approximations of peak points by spreading them over a region to give the illusion of continuity in transition.
- Dithering can be in
    1. *Audio* to produce smooth waveforms when down sampling from digital to analog.
    2. *Image* for smoothing out banding and to make images ready for printing on a dot format printer/screen.

IMAGE DITHERING

- Sometimes also referred to as Digital Halftoning or Spatial Dithering.
- *Dithering used in computer graphics to create the illusion of "color depth" in images with a limited color palette - a technique also known as color quantization. In a dithered image, colors that are not available in the palette are approximated by a diffusion of colored pixels from within the available palette. The human eye perceives the diffusion as a mixture of the colors within it (see color vision)".*

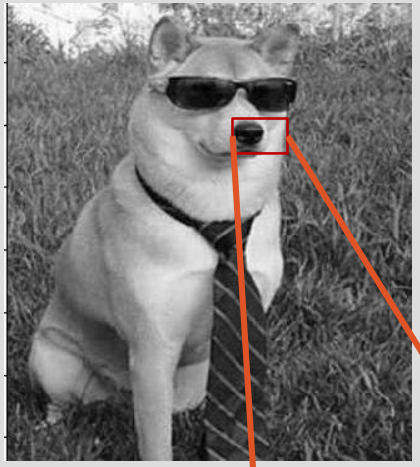
[Wikipedia]

IMAGE DITHERING (CONTD.)

- Most digital images are of 8-bit color depth i.e. each channel (for RGB) has a value between 0 and 255 (256 color levels) and for 256 gray levels in black-and-white.
- Printers only have red-green-blue and black ink available and no in between shades, so to translate these 8-bit images to be printable these colors are approximated as a binary values of put ink or don't put ink.

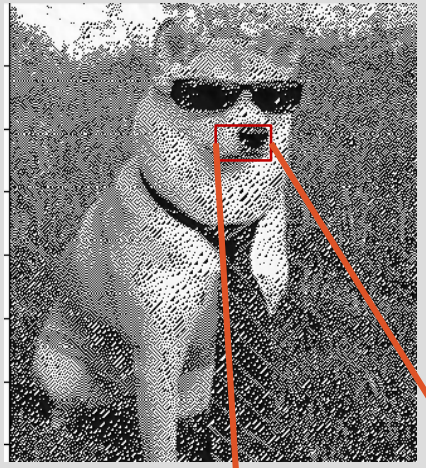
Color Doggo

B/W Doggo



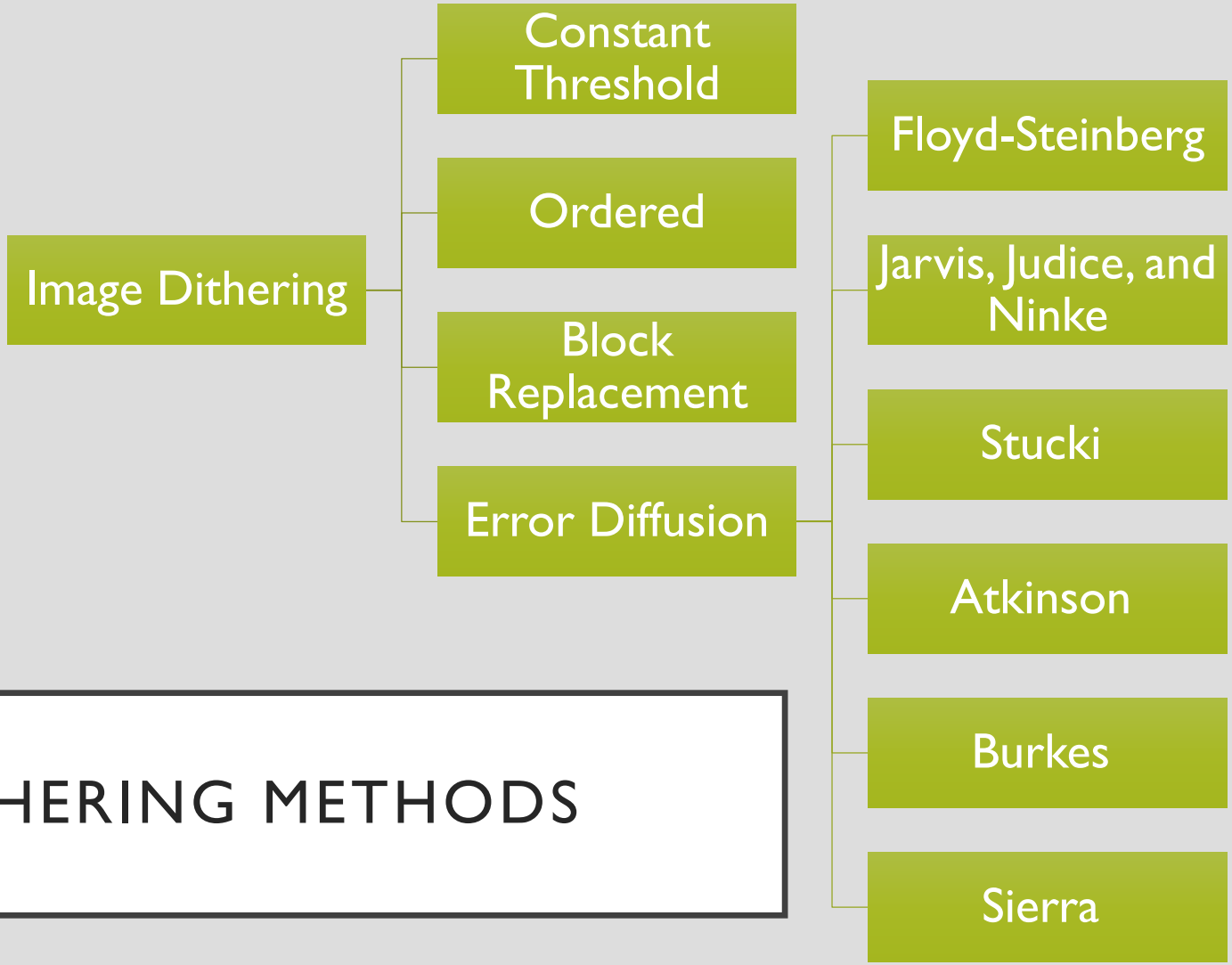
snoot

Dithered Doggo



snoot

DITHERING METHODS



BRIEF OVERVIEW OF OTHER DITHERING ALGORITHMS

CONSTANT THRESHOLDING

- This is the simplest and most straight forward of the methods. it thresholds the pixels based on a threshold (average, 127, mode).
- Complexity: $O(N^2)$.
- The most optimal parallelization is achieved using n processors where each processor has one row. This reduces the complexity to O(n) and cost of $O(n^2)$.

```
threshold (image, th){
                    for each pixel Pi,j  in image{
                                Pi,j = 0 if Pi,j < th else 255
                    }
    }
```

Serial Constant Threshold
using global mean

Parallel Constant Threshold
using local mean

ORDERED DITHERING

- Provides a fixed pattern of numbers to indicate the order of turning pixels on within a "screen" before they're turned to binary.
- A threshold matrix also called the screen is provide and overlaid on the image through replication
- Complexity: $O(N^2)$.

```
Ordered_dither ( I, Th) {
    For each pixel  Pi,j {
                Pi,j = 1  if Pi,j > T[i%R, j%C] else 0
    }
}
```

PARALLELIZATION OF ORDERED DITHERING

- Uses N processors where N is the number of rows in the Image.
- Each processor gets one row of both the image and the threshold matrix.
- This reduces the complexity to O(N).
- Speed up of O(N) is observed.
- This approach is cost optimal for images of size > 800 x 600.

Serial ordered

Parallel ordered

BLOCK REPLACEMENT

- each pixel in the original image is replaced by one of a predetermined set of binary patterns (i.e. matrices). The dimension of the patterns is determined by screen frequency and the print resolution.
- The replacement is based on matching grey level of a pixel in a set of predefined matrices.
- There's 2X2 matrices for faster matching but they provide poor results thus 3X3 matrices are used.

BLOCK REPLACEMENT SERIAL/PARALLEL

- Due to the inner working of the algorithm, there is no visible difference between the outputs of the serial and parallel methods.

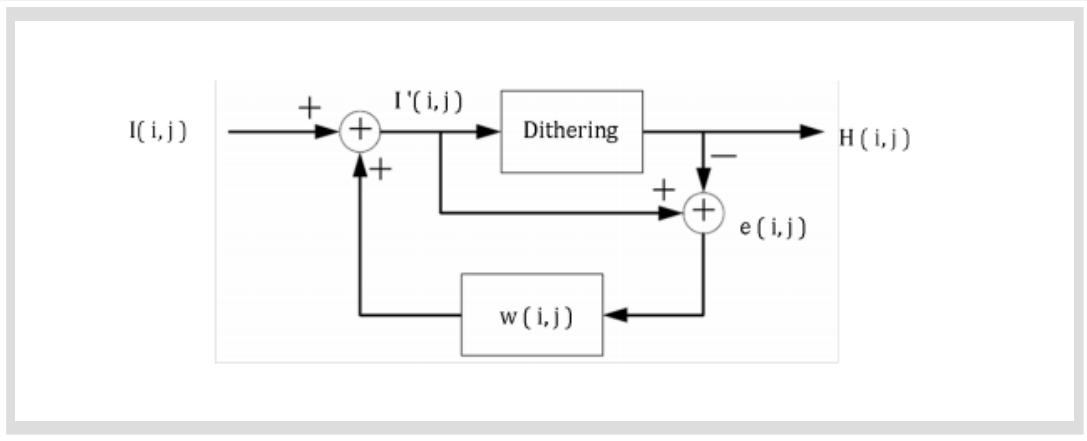
- Img source : Digital Halftoning, Report 2011 Sadan Ekdemir, Xunxun Wu Uppsala University

ERROR DIFFUSION

- The quantization residual is distributed to neighboring pixels that have not yet been processed.
- Converts a multi-level image into a binary image.
- Error diffusion is classified as an area operation – activity in one location influences what happens at other locations.
- Tendency to enhance edges in an image, giving a crisper output than other dithering techniques.

ERROR DIFFUSION DITHERING

- Unlike other methods, which treat each pixel individually, error diffusion quantifies each pixel using a neighbourhood operation.
- This method moves through the original image in raster order, normally starting from the pixel up to the left (i.e. the first element of the matrix) and then goes through all pixels from left to right until the end.



METHODS

Error Diffusion

Floyd- Steinberg

	i	7/16
3/16	5/16	1/16

Jarvis, Judice, and Ninke

		1	8/42	4/42
2/42	4/42	8/42	4/42	2/42
1/42	2/42	4/42	2/42	1/42

Stucki

		1	7/48	5/48
3/48	5/48	7/48	5/48	3/48
1/48	3/48	5/48	3/48	1/48

Atkinson

		i	8/32	4/32
2/32	4/32	8/32	4/32	2/32

Burkes

	1	1/8	1/8	
1/8	1/8	1/8		
	1/8			

Seirra

		1	5/32	3/32
2/32	4/32	5/32	4/32	2/32
	2/32	3/32	3/32	

PARALLELIZATION IN ERROR DIFFUSION DITHERING

And other such jokes you can tell yourself.

THE PROBLEM IN PARALLELIZING.

- The algorithm cannot be directly parallelized due to data dependencies.
- Value of a pixel is determined not only by its own value but also the diffused error.
- A pixel at location $image[i, j]$ depends on the values from $\{ Image_{i-1,j}, Image_{i-1,j-1}, Image_{i-1,j-1}, Image_{i+1,j-1} \}$
- Without these dependencies solved the produced output would be some intermediate with out of order updates.

ENOUGH THEORY, LET'S PARALLELIZE.

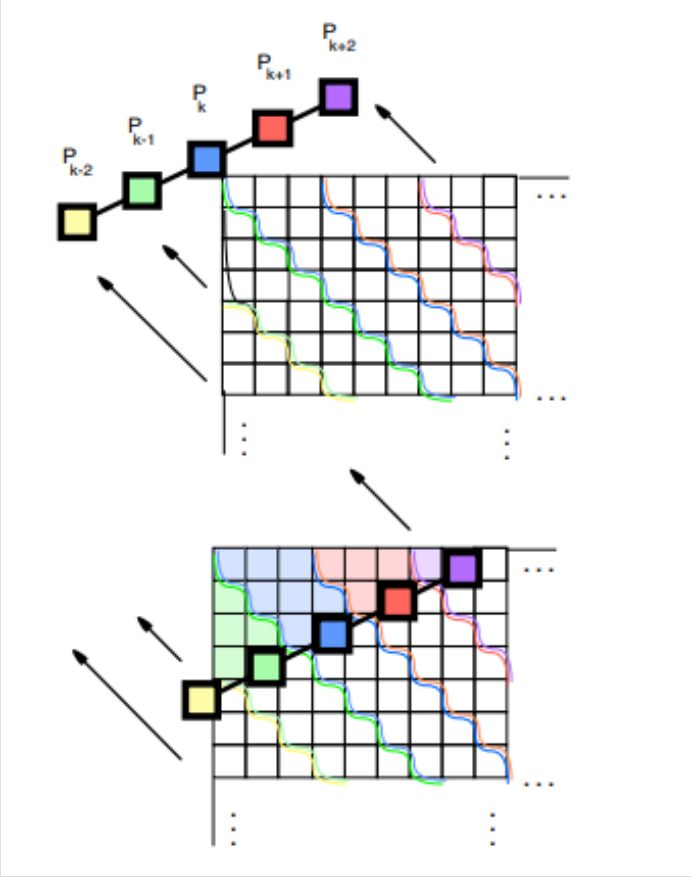
OPTIMAL SCHEDULING IN A LINEAR ARRAY

- A linear array of P processors is used which work on an optimum scheduling strategy.
- Whenever two pixels have equal scheduling times they can be processed simultaneously.
- P_i has row i and when its done diffusing errors for row i+ 1 for the data-dependency region, it makes a call to P_{i+1} to start.
- For P_{i+1} to start, along with the call from P_i, it also checks the following:

$$Img(i,j) = 1 + \max(Img[i-1][j], img[i-1][j-1], img[i][j-1], img[i+1][j-1])$$

- All Values in max must be true.

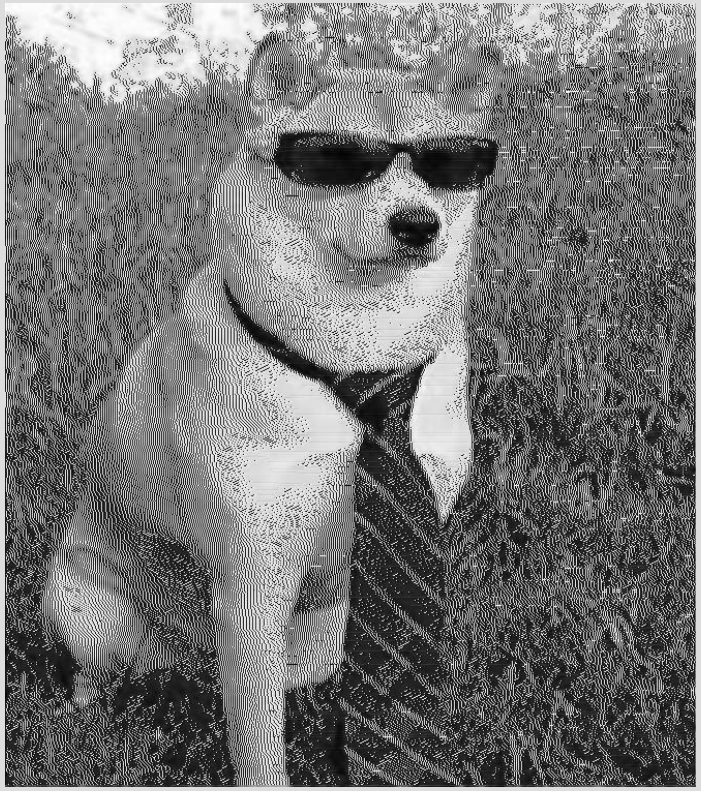
MOVEMENT OF PROCESSORS



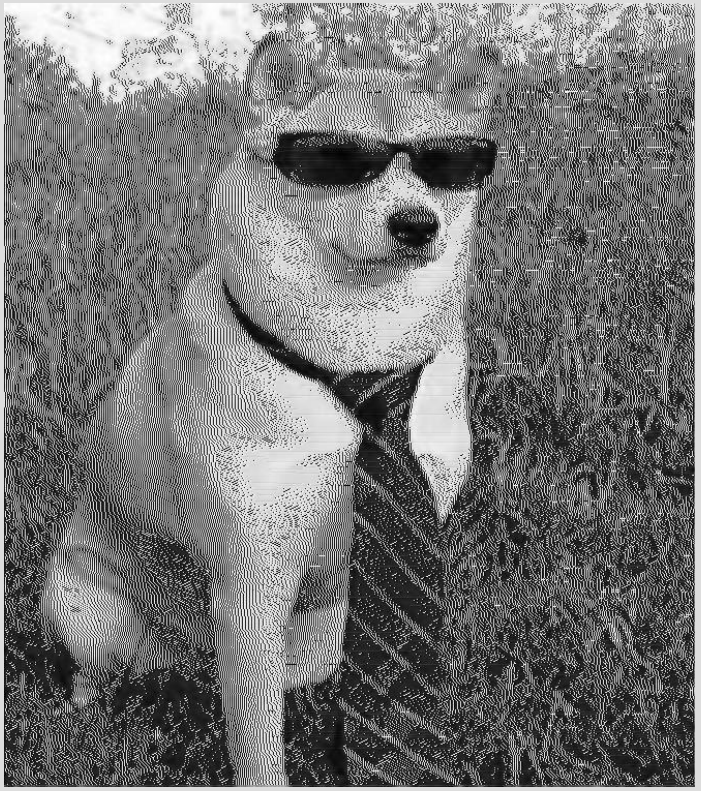
ANALYSIS

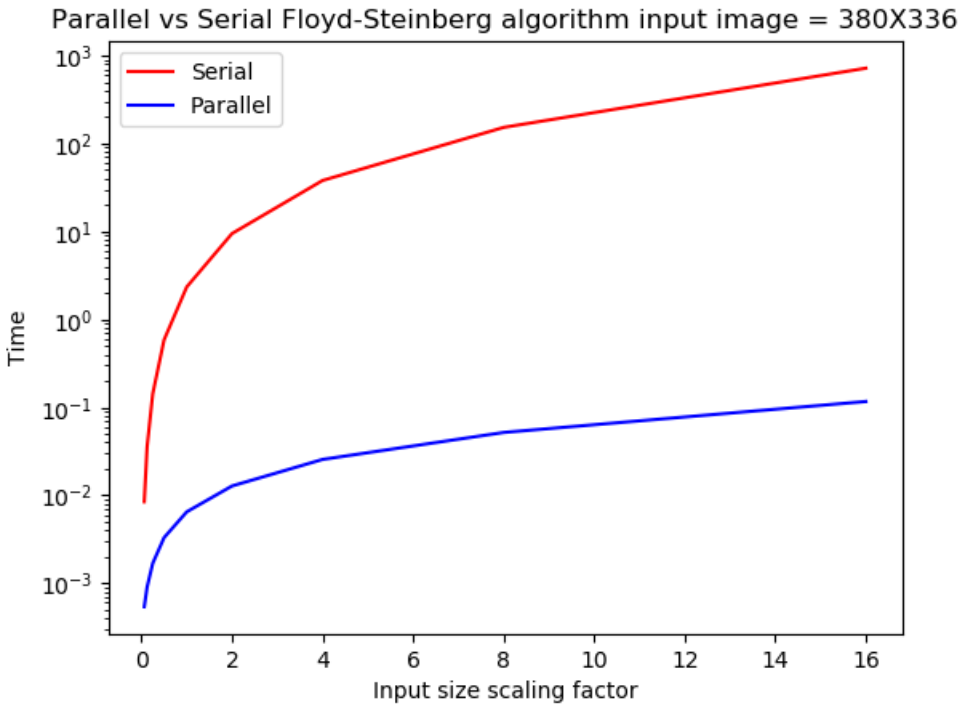
- The running time of the algorithm is O(2n + m). Whereas the normal algorithm takes O(NM). Where n and m are the number of rows and columns in the image.
- It is rare to see a gain of this order due to the overhead involved in message passing.

SERIAL IMAGE



PARALLEL IMAGE



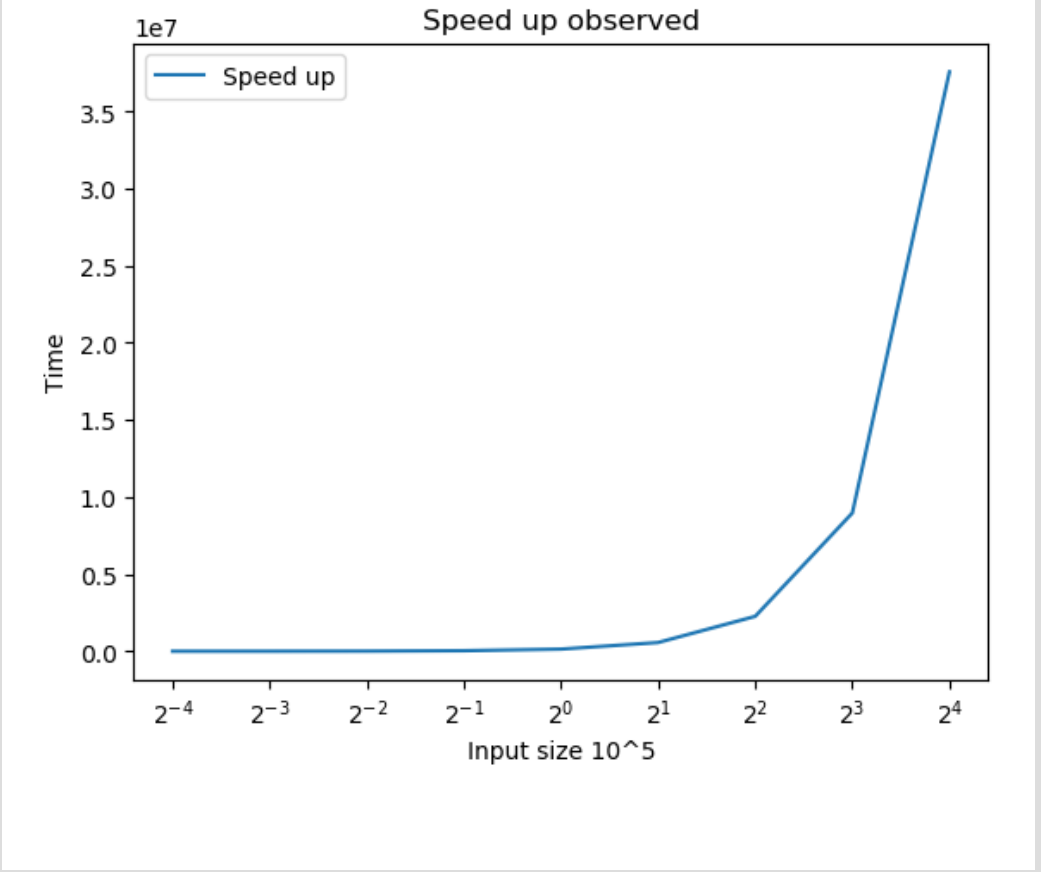


COMPARISION

- The scale on Y axis is logarithmic.
- Parallel Algorithm shows a slower growth rate on higher orders of input.

SPEED-UP

- The raw speed-up is in the accompanying graph.
- The expected speedup is of the order on max(N, M)

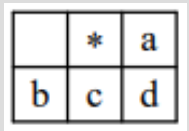


CHECKER-BOARDING

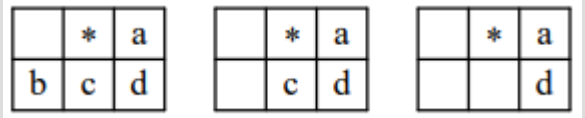
- This method of parallelization uses the spiral pinwheel method of error diffusion.
- The entire image is divided into an alternating pattern of ‘Grey’ and ‘white’ blocks also called as inward and outward blocks.
- Each block is assigned to a single processor.
- The inward method uses 3 weight matrices while the inward just uses one.

DIFFUSION PATTERNS

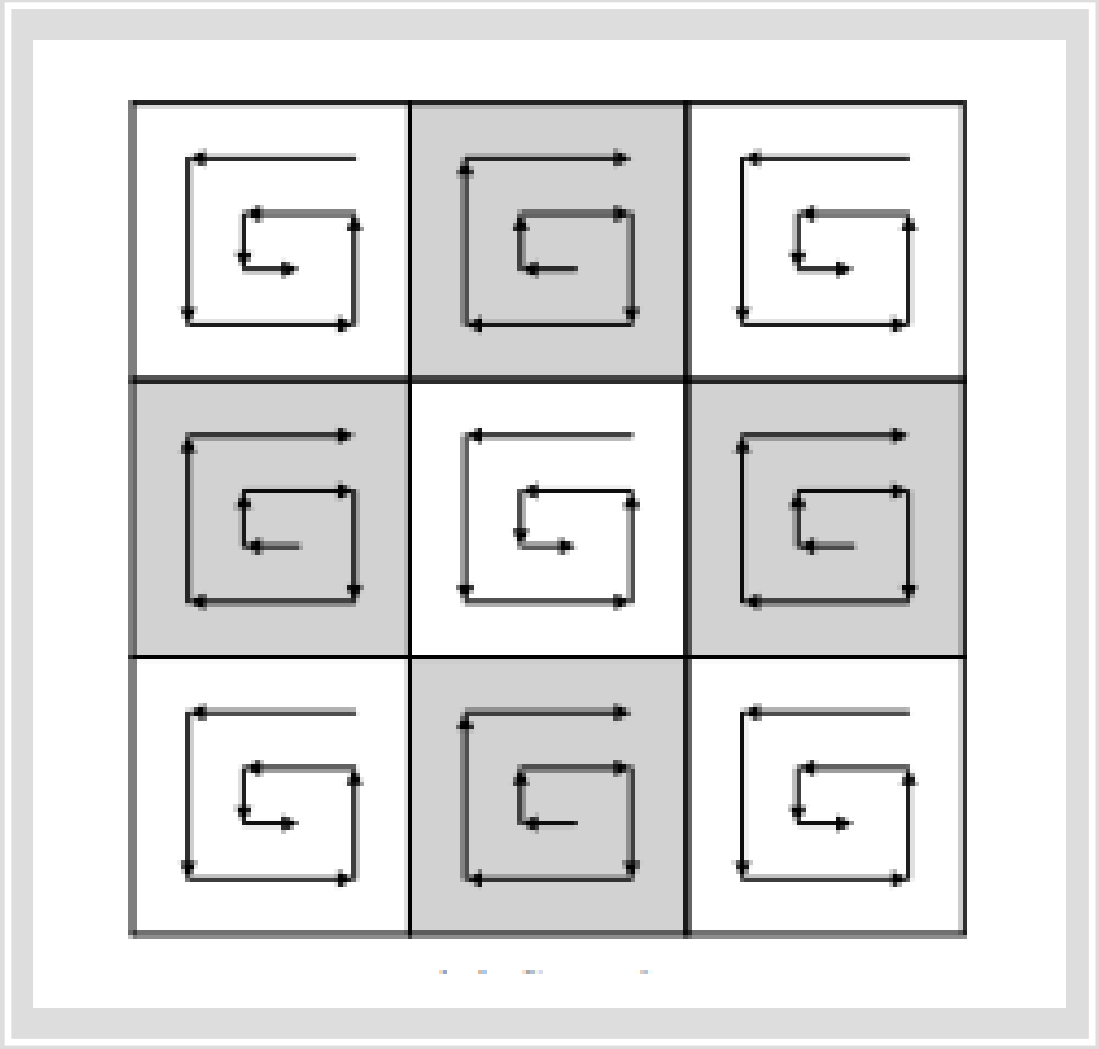
- The grey blocks are the outward blocks.
- The white blocks are the inward blocks.

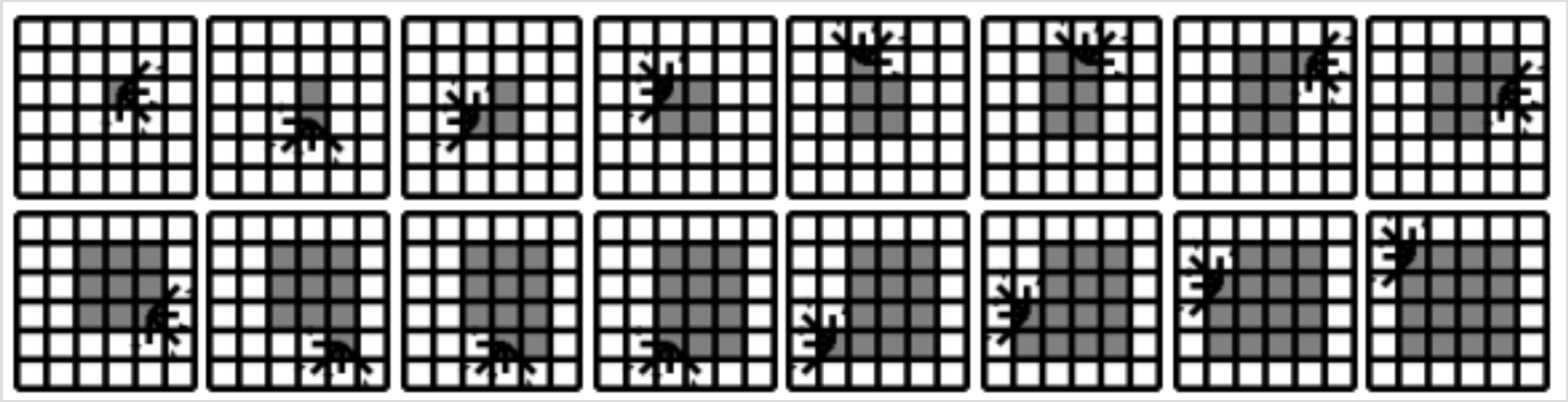


Outward matrix



Inward matrix

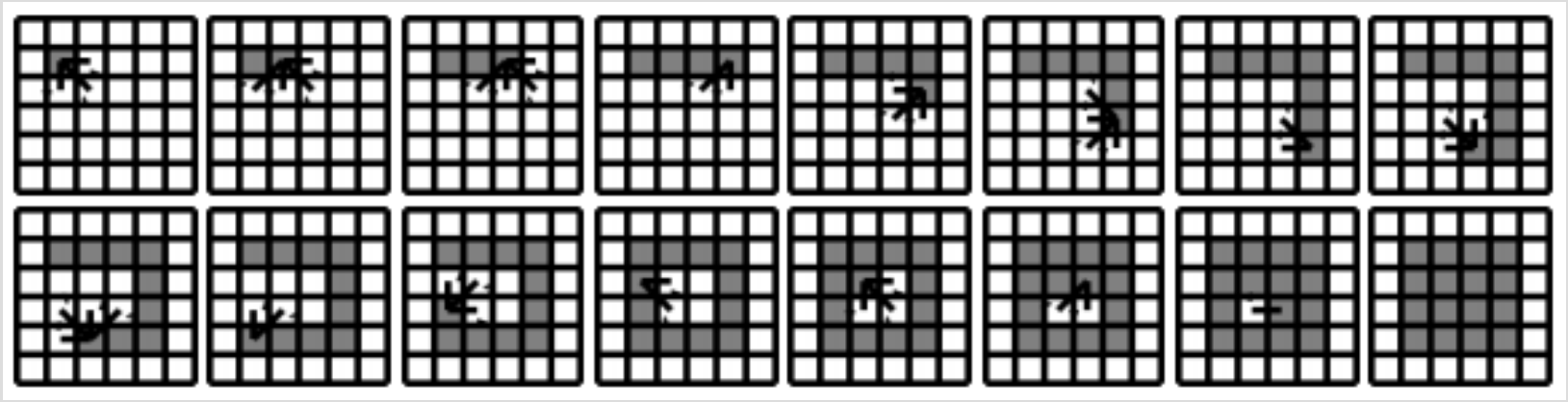




Outward matrix

ADDRESSING PATTERNS

Inward matrix

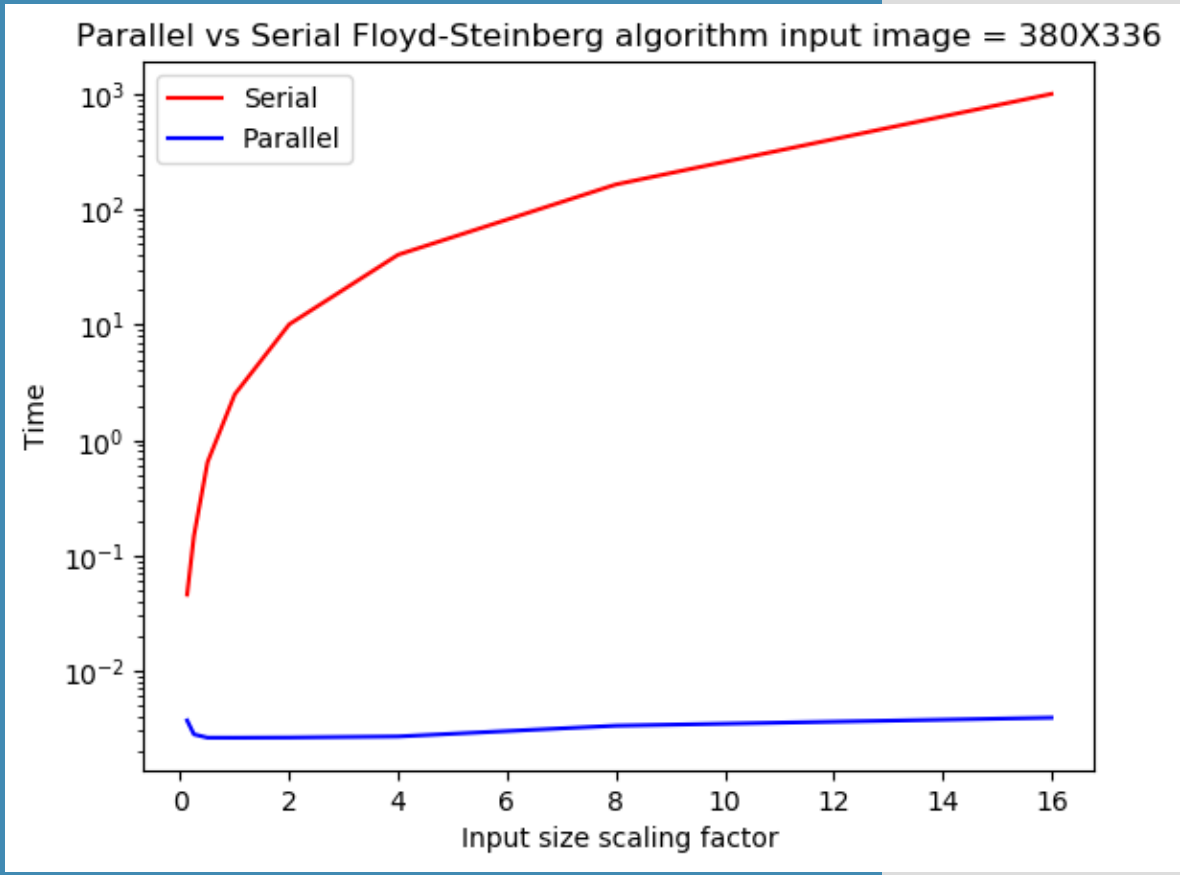


SALIENT FEATURES

- The main feature here is the matrices are rotated based on the direction of motion of the pixel iterators.
- This ensures that the errors don't diffuse to out of bound regions or over-write dithered pixels.
- This logic of rotating the matrices made this approach possible because under normal conditions the serpentine transversal would've done over-writes.

ANALYSIS

- The theoretical max throughput is given by :
  $No.Processors * No.Theads * Processor\ Frequency.$
- The algorithm is of the order O(block_size^2 * num_blocks)

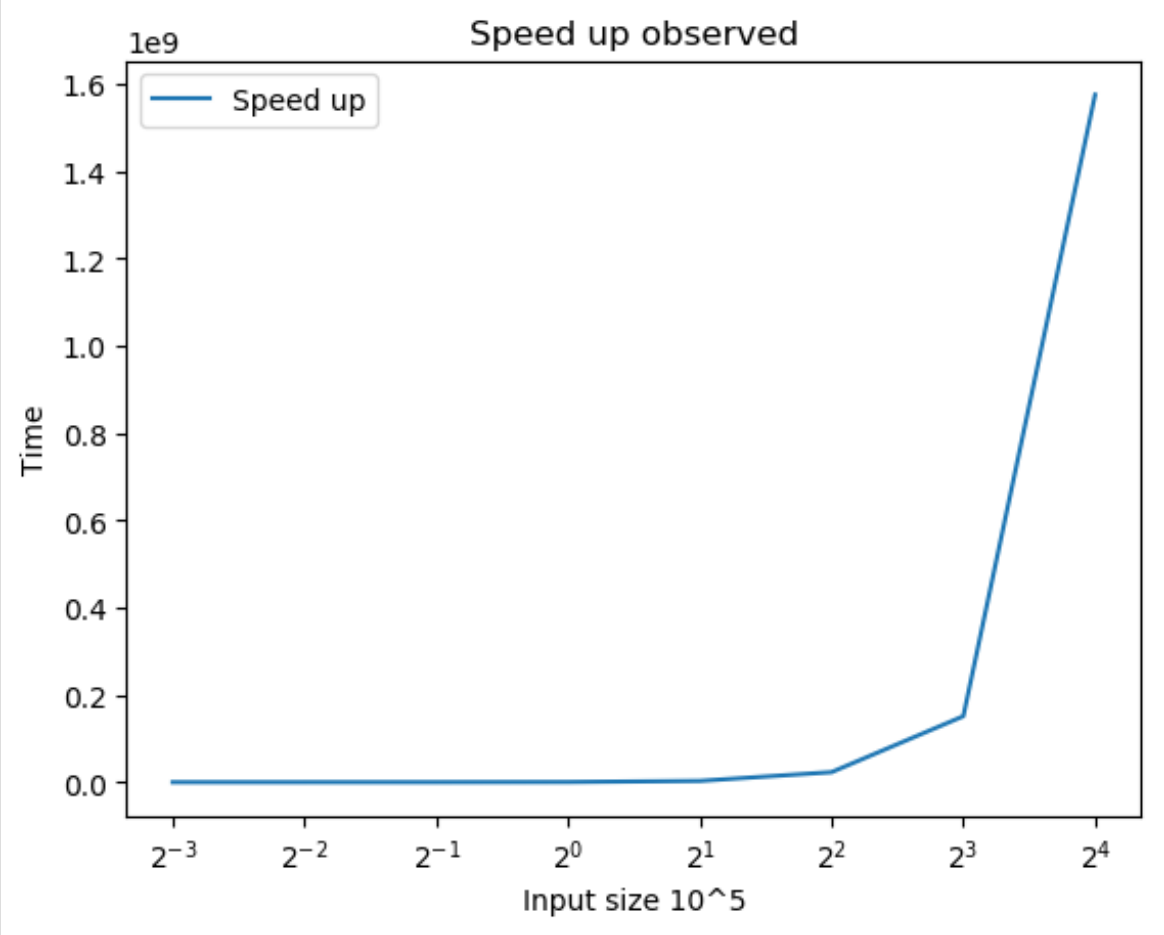


COMPARISION

- The scale on Y axis is logarithmic.
- Parallel Algorithm shows a slower growth rate on higher orders of input.

SPEED-UP

- The raw speed-up is in the accompanying graph.
- The expected speedup is of the order on max(N, M)



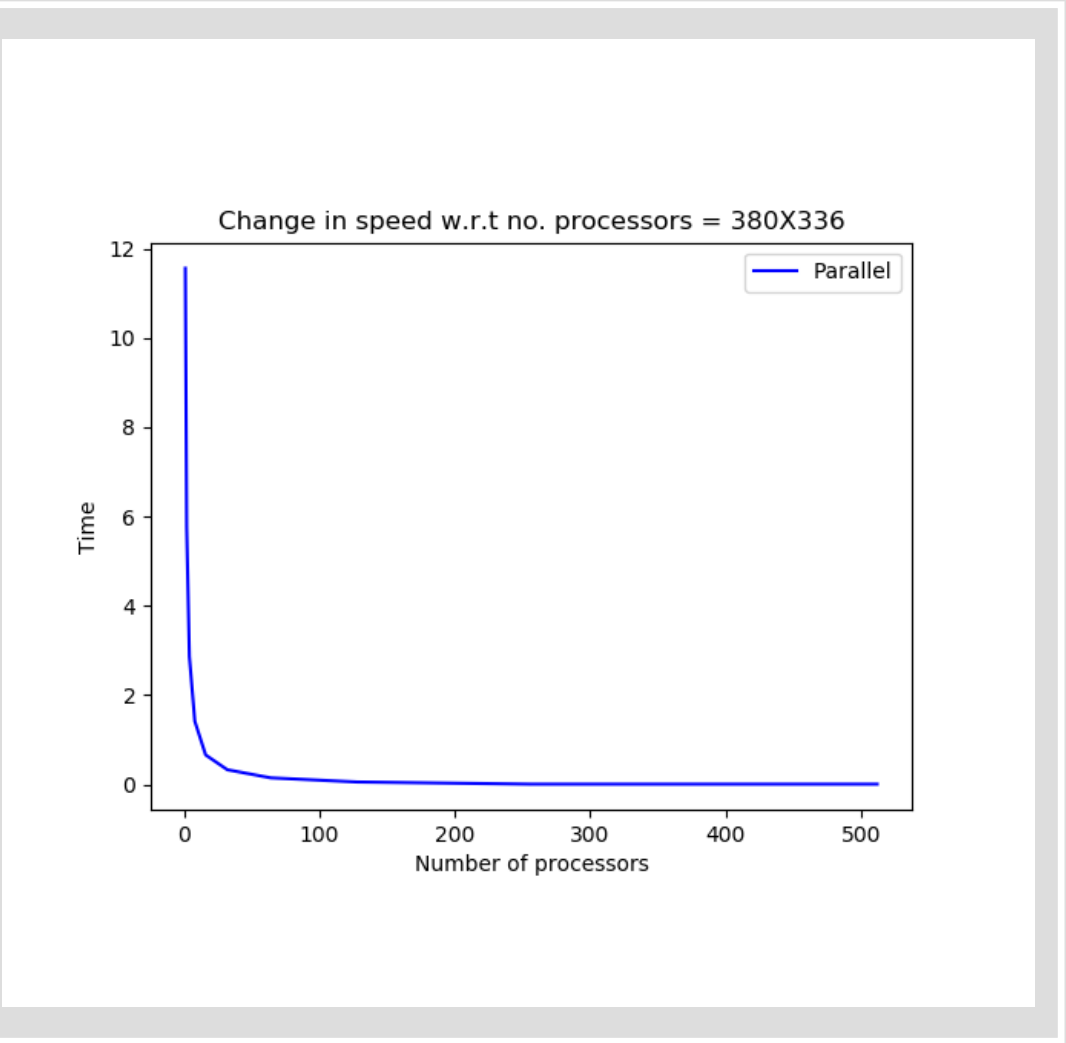
MY APPROACH*

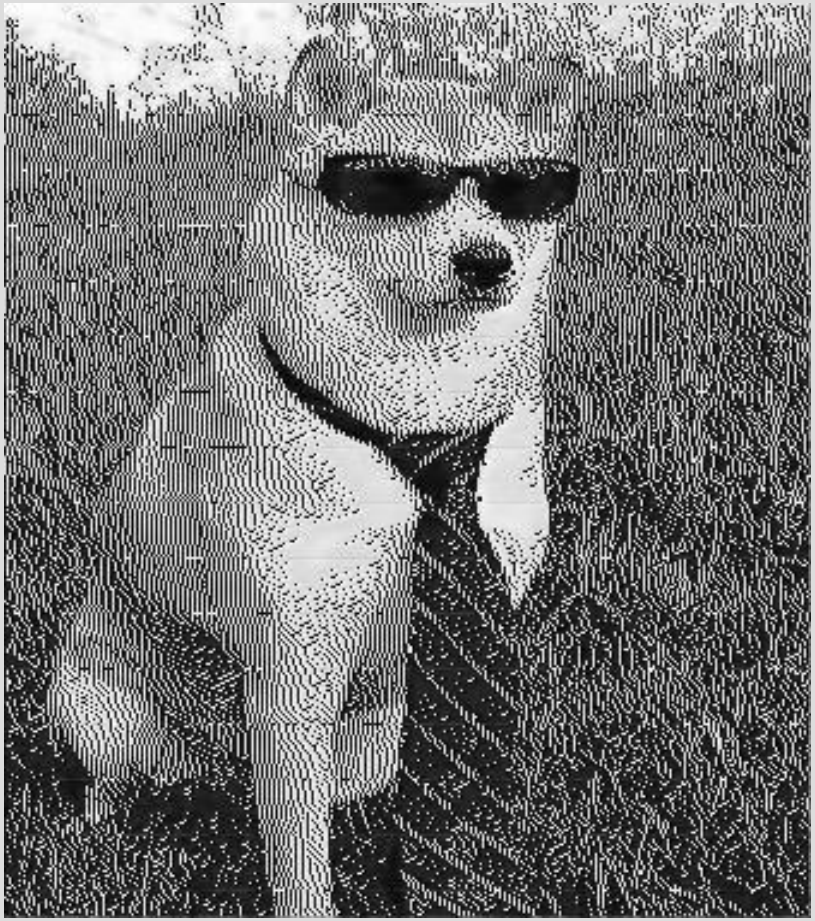
- The image is divided in N blocks, where N is the number of processors.
- Each processor gets a block of size N/num_procs+1 and it treats it as an independent image.
- The processing is done in the range 0 to N/num_procs while this N/num_procs+1^{th} row is for diffusion only .
- This approach can be said to a variant of the Optimal Scheduling method but with restricted processors.

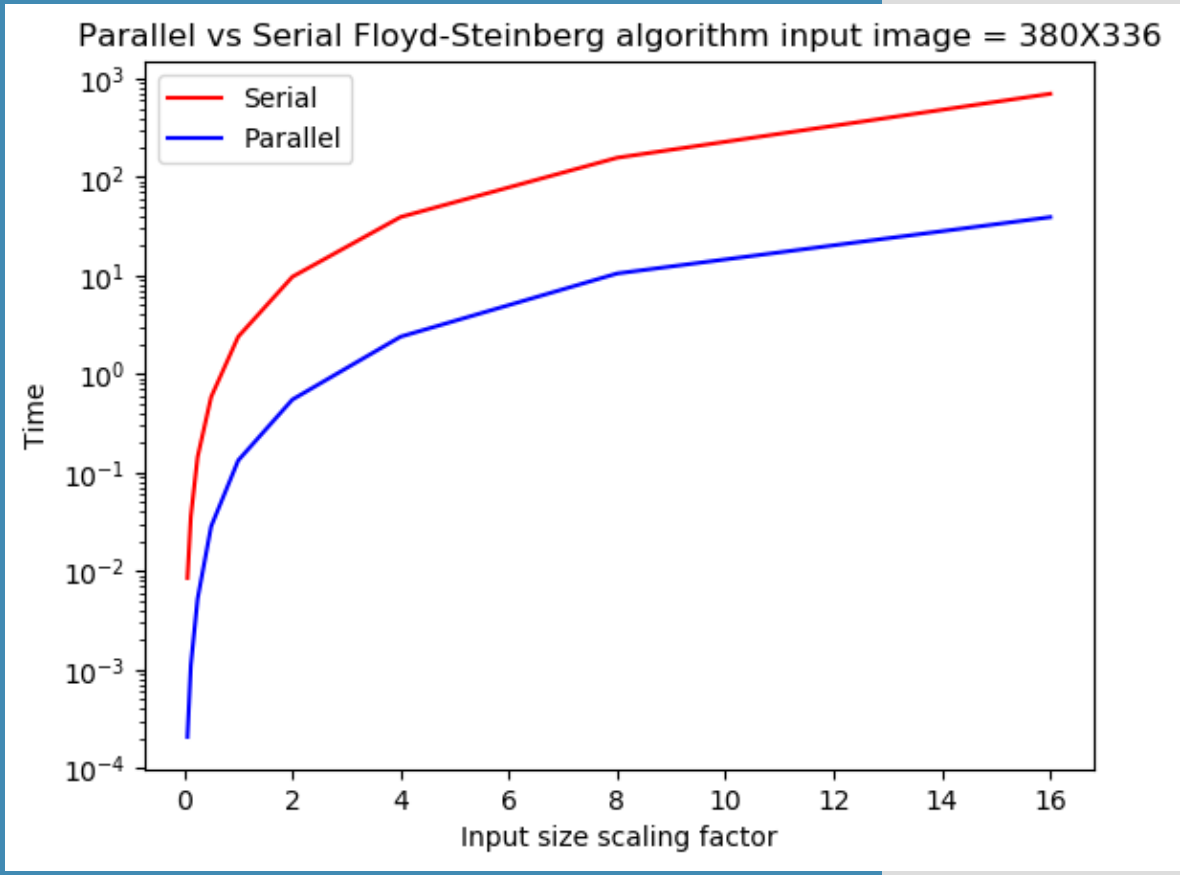
* Mostly based on the algorithm by **Digital Halftoning, Report 2011 Sadan Ekdemir, Xunxun Wu Uppsala University**

DIVISION OF IMAGE

- The complexity here completely depends on the number of processors as it is evident from the graph.
- The optimal for the given image size was found to be 16 processors after which there was only negligible change in time.





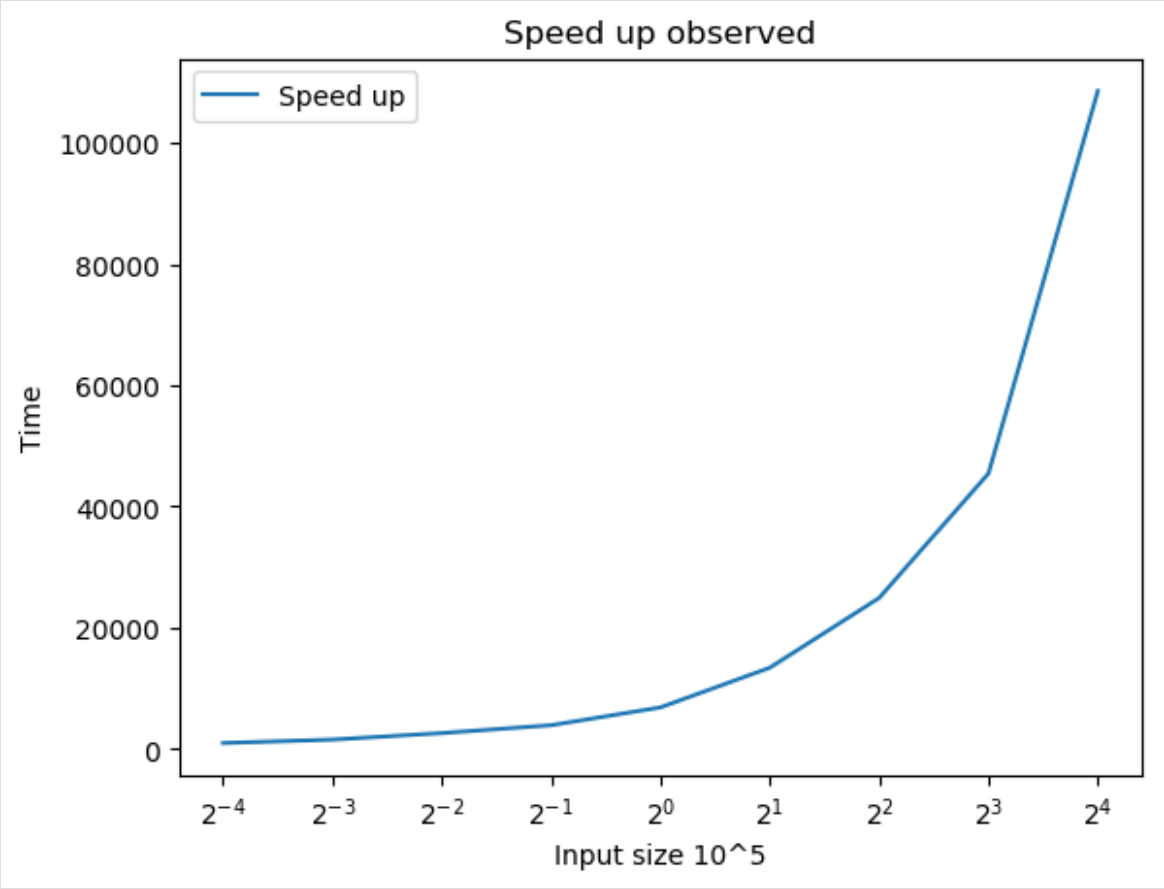


COMPARISION

- The scale on Y axis is logarithmic.
- Though both grow similarly, in each tested case the overall time was reduced by a factor of N.

SPEED-UP

- The raw speed-up is in the accompanying graph.
- The speed-up is again a factor of the number of processors, with higher number of processors showing better speed up.



REFERENCES

- Parallel Digital Halftoning by Error-Diffusion, Panagiotis Metaxas. Principles of Computing & Knowledge, Paris C. Kanellakis Memorial Workshop, San Diego, California, USA, June 8th, 2003
- A Parallel Error Diffusion Implémentation on a GPU, Zhang et al. Proc. SPIE 7872, Parallel Processing for Imaging Applications, 78720K (25 January 2011)
- Lee Crocker, Image Dithering.
- Tanner Helland, Image Dithering: Eleven Algorithms and Source Code
- All codes related to this are available on https://github.com/IsCoelacanth/PA_Assignment