



Parallelizing Sobel Edge Detection and Gaussian Blur

Sumanth N Hegde - 191ME284
Saransh Bhaduka - 191ME175
Pratham N - 191MT034

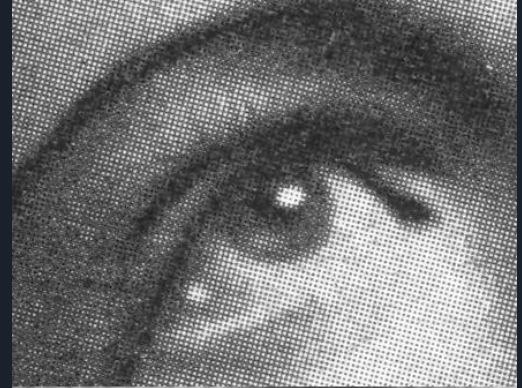



An Introduction to Gaussian Blur

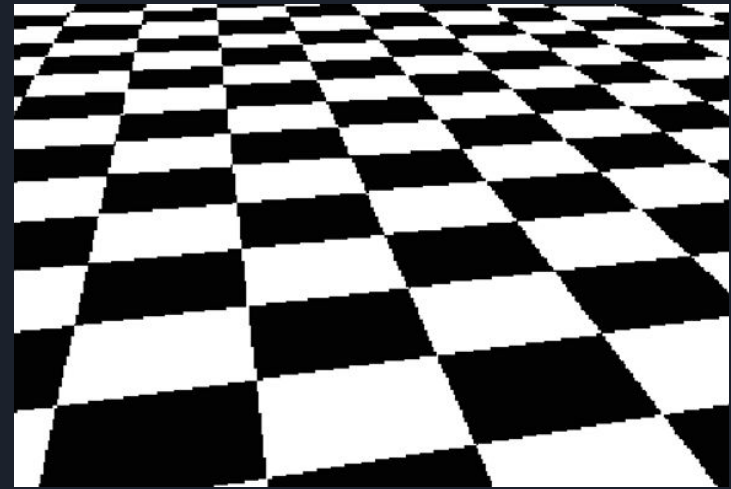
- In image processing, a Gaussian blur (also known as Gaussian smoothing) is the result of blurring an image by a Gaussian function (named after mathematician and scientist Carl Friedrich Gauss).
- It is a widely used effect in graphics software, typically to reduce image noise and reduce detail.
- Gaussian smoothing is also used as a pre-processing stage in computer vision algorithms in order to enhance image structures at different scales.
- A type of low-pass filter, Gaussian blur smoothes uneven pixel values in an image by cutting out the extreme outliers.

Uses of Gaussian Blurring

- A halftone image can be rendered smooth and better looking using Gaussian Blur, as shown in the figure.
- If you take a smartphone photo in low light and the resulting image has a lot of noise, Gaussian blur can mute that noise.



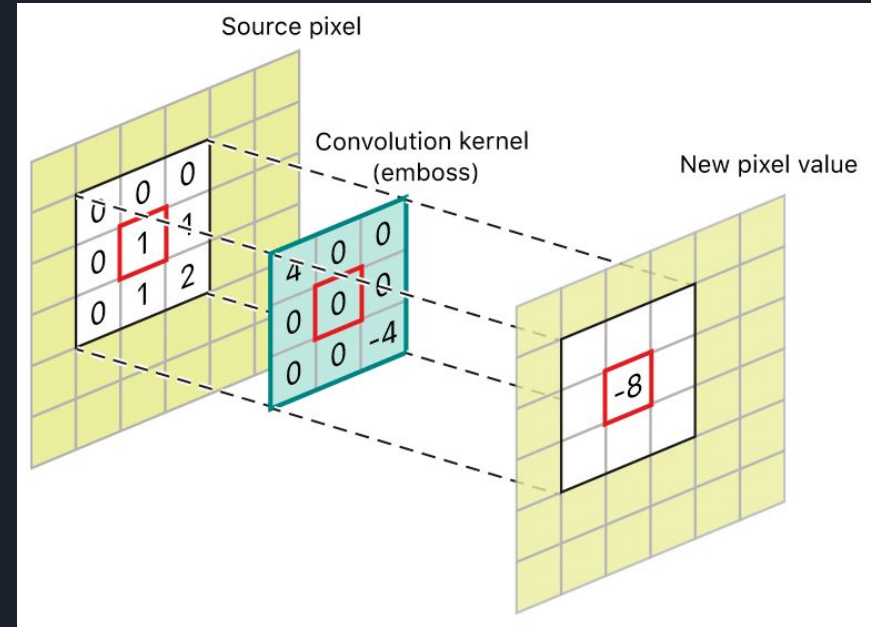
- 
- The formation of moiré pattern aliasing artifacts can be avoided as well during Gaussian Blurring. Even though the image is blurred, the aliasing is avoided. This is seen in the figure.
 - Gaussian smoothing is commonly used with edge detection. Most edge-detection algorithms are sensitive to noise.
 - Using a Gaussian Blur filter before edge detection aims to reduce the level of noise in the image, which improves the result of the following edge-detection algorithm.



How it works- Convolution

In simple terms, convolution is simply the process of taking a small matrix called the kernel and running it over all the pixels in an image. At every pixel, we'll perform some math operation involving the values in the convolution matrix and the values of a pixel and its surroundings to determine the value for a pixel in the output image.

By changing the values in the kernel, we can change the effect on the image — blurring, sharpening, edge detection, noise reduction, etc.





Gaussian Formula

The weight of main pixel and the other weights of less weighted pixels can be calculated as in (1) Gaussian equation

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{\frac{-x^2+y^2}{2\sigma^2}}$$

In this equation the parameters are explained as follows:

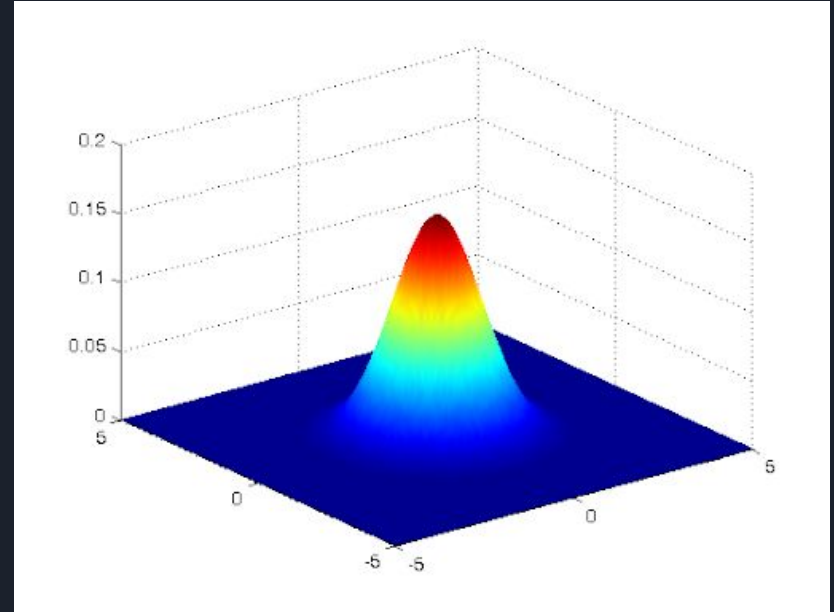
- σ : blur factor: If this value increase, image will blur.
- e : euler number
- x : Horizontal distance to centre pixel
- y : Vertical distance to centre pixel


According to this equation, x and y distances will be zero for centre pixel. When distance increases from centre pixel x^2+y^2 value will increase and weights will decrease. If this formula is applied to two dimensions, bell shape distribution appears from contour centre point and forms homocentric circle surfaces. These distribution values are used for making a convolution matrix. That matrix is applied to original image. Every new value of pixels can be computed by weighted averaging of itself and neighbour pixels. Centre pixel takes highest average weight. Nonetheless, the weight of the neighbour pixels is directly proportional to the distance to the centre. This process comes to a conclusion with a blur, conserves borders and edges.

Gaussian Distribution

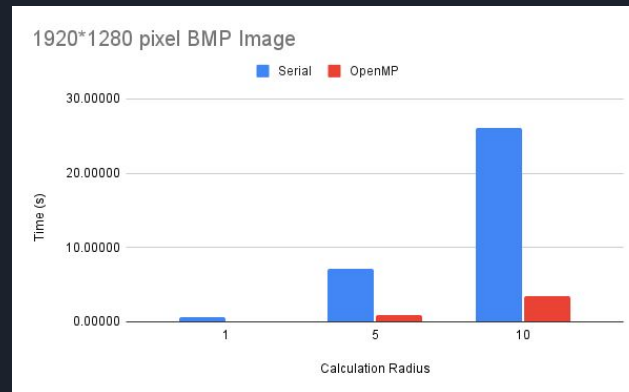
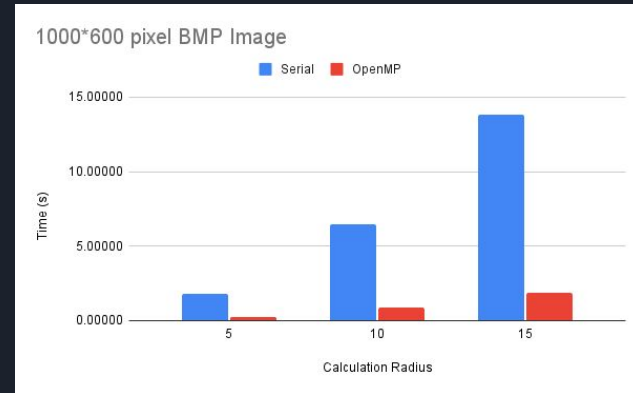
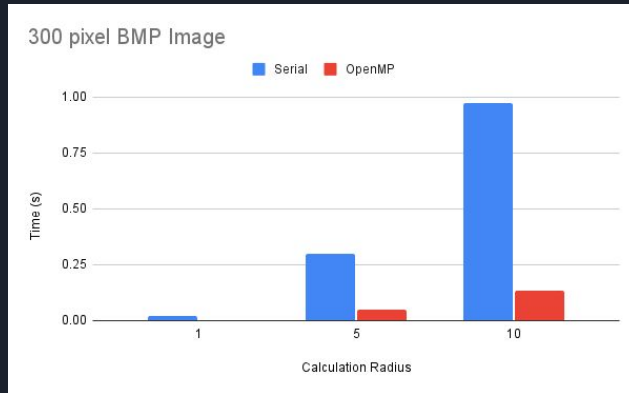
Imagine that this distribution is superimposed over a group of pixels in an image. It should be apparent looking at this graph, that if we took a weighted average of the pixel values and the height of the curve at that point, the pixels in the center of the group would contribute most significantly to the resulting value. This is, in essence, how Gaussian blur works.

We will take the Gaussian function and we'll generate an $n \times m$ matrix. Using this matrix and the height of the Gaussian distribution at that pixel location, we'll compute new RGB values for the blurred image

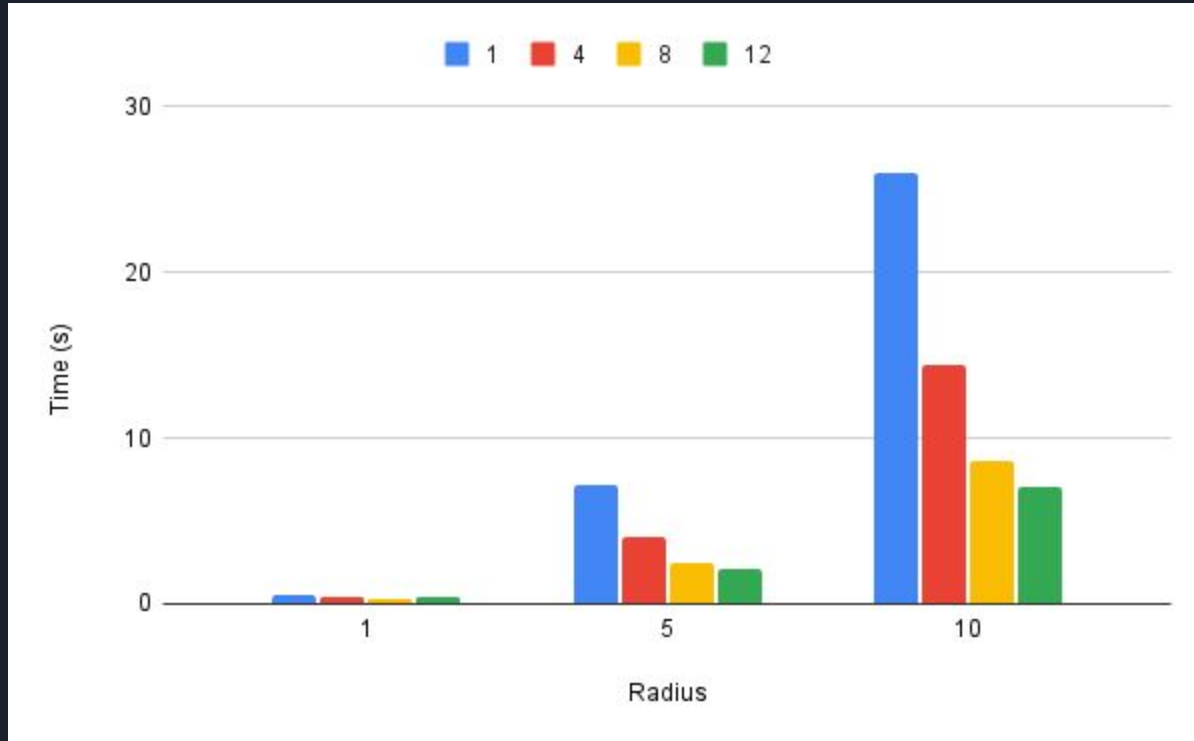


- 
- Edge cases are straightforward too. Where an edge pixel is sampled, the otherwise non-existent surrounding pixels are either given the same value of their nearest neighbor, or given a value matching up with their mirror opposite pixel in the sampled area.
 - The same calculation is run for each pixel in the original image to be blurred, with the final output image made up of the pixel values calculated through the process. For grayscale images, it's that simple. Color images can be done the same way, with the blur calculated separately for the red, green, and blue values of each pixel. Alternatively, you can specify the pixel values in some other color space and smooth them there.

Serial vs OpenMP



MPI performance for various number of processors- 1920*1280 24-bit BMP image

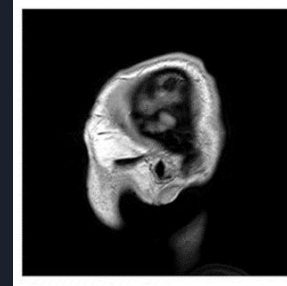
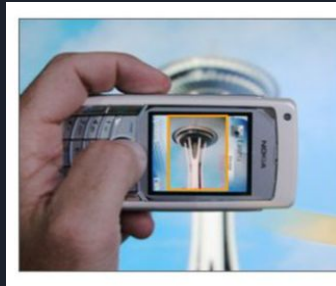
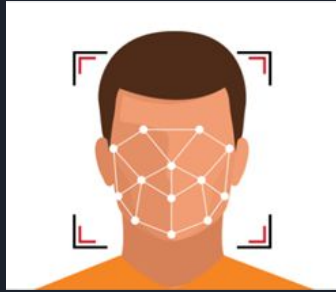




An Introduction To Edge Detection

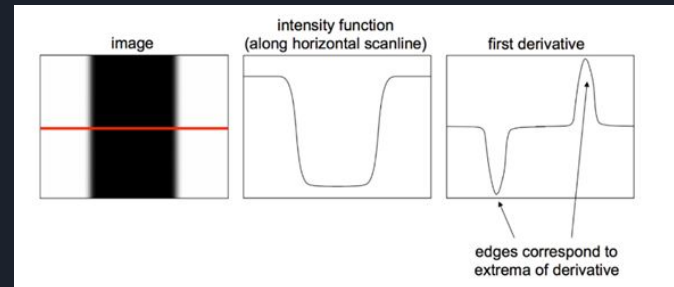
- Edge detection is a widely used image segmentation technique that identifies edges in an image.
- The edges are also represented as a collection of interconnected points located at the border between the two areas. The edges illustrate the borders of the object, and hence this feature can be used to segment an image toward its primary areas or objects.
- Canny, Prewitt and Sobel are the most commonly used algorithm in the edge detection.

Applications



Sobel Edge Detection

- Presented as a discrete differential operator technique for gradient approximation computation of the image intensity function, Sobel edge detection enables low-level feature extraction and dimensionality reduction.
- Edges are high frequency content. In edge detection, we want to retain these edges and discard everything else. hence we should build a kernel that is equivalent of a high pass filter.



Sobel Edge Detection

- Create kernel 3x3 matrix -> Kernels convolve over image -> Matrix multiplication to calculate the intensity value -> Gradient component combined to find absolute magnitude at each point and the orientation of the gradient.
- Sobel operators give the edges where intensity changes faster.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

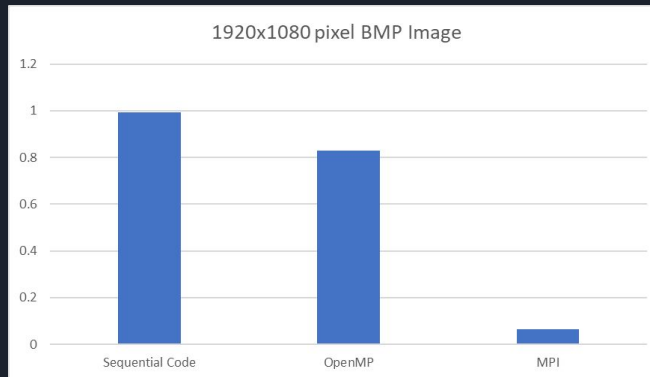
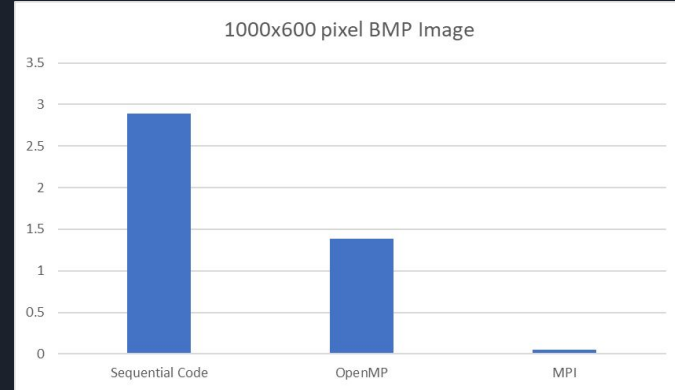
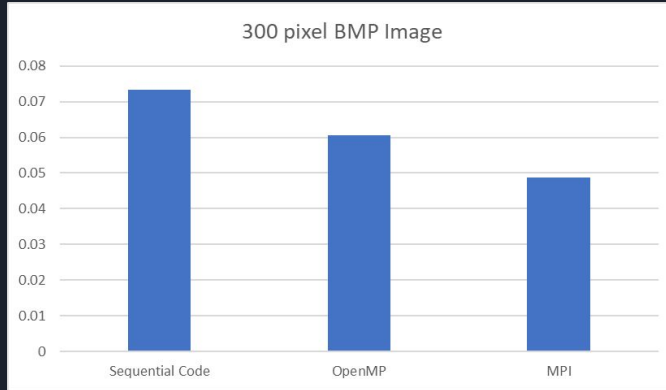
100	100	200	200
100	100	200	200
100	100	200	200
100	100	200	200

-1	0	1
-2	0	2
-1	0	1

-100
-200
-100
200
400
+200
=400

Kernel Convolution: The bigger the value at the end, the more noticeable the edge will be.

Serial vs OpenMP vs MPI





Results and Conclusions

- It is certainly evident that parallel multicore Sobel and Gaussian algorithms improves the performance of the traditional sequential Sobel and Gaussian algorithms by fully utilize the CPU to its utmost potential.
- This work is not limited to image processing methods only but can be extended to other processor intensive application.



Thank You