Image Convolution

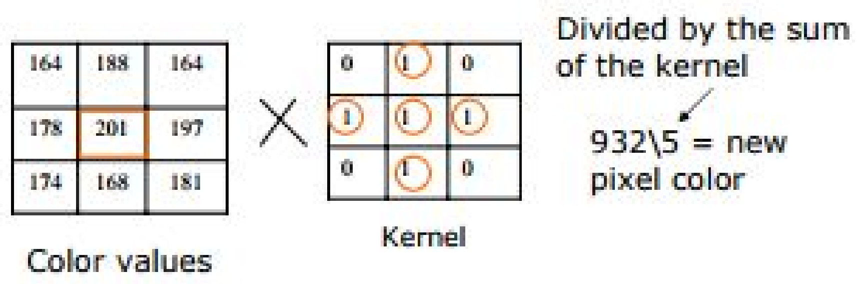
2D Convolution of a given image with a predefined matrix to get effects such as blur, sharpen, emboss and sobel.

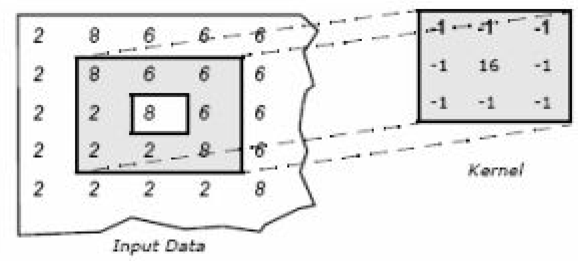
201501177 - Naitik Dodia
201501219 - Kaushal Patel

Algorithm

Input: Image and kernel

Output: Convoluted image





Algorithmic Complexity

- Let n = pixels in a row and m = height of the kernel
- For each pixel
  - Computations = m^2
- Total time complexity: $O(n^2 * m^2)$

Parallel Time: There is no dependency in the computation part so all the computation part can be divided into available processors.

Parallel time complexity: $O((n^2 * m^2)/p)$

Kernels used

Sharpen

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Blur

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Emboss

$$\begin{pmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & -1 & 2 \end{pmatrix}$$

Sobel

$$\begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & 2 \\ 1 & 0 & -1 \end{pmatrix}$$

Original Image

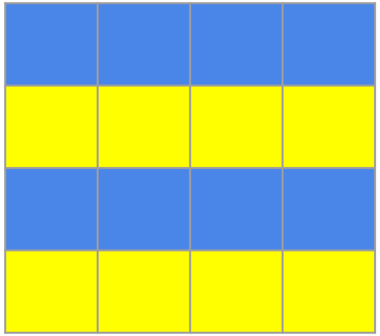
Sobel Kernel

Sharpen Kernel

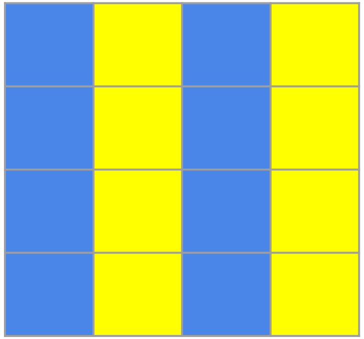
Emboss Kernel

Parallelization Techniques

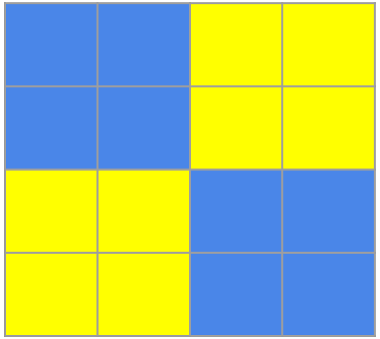
Row Wise Parallel



Column Wise Parallel



Block wise parallel

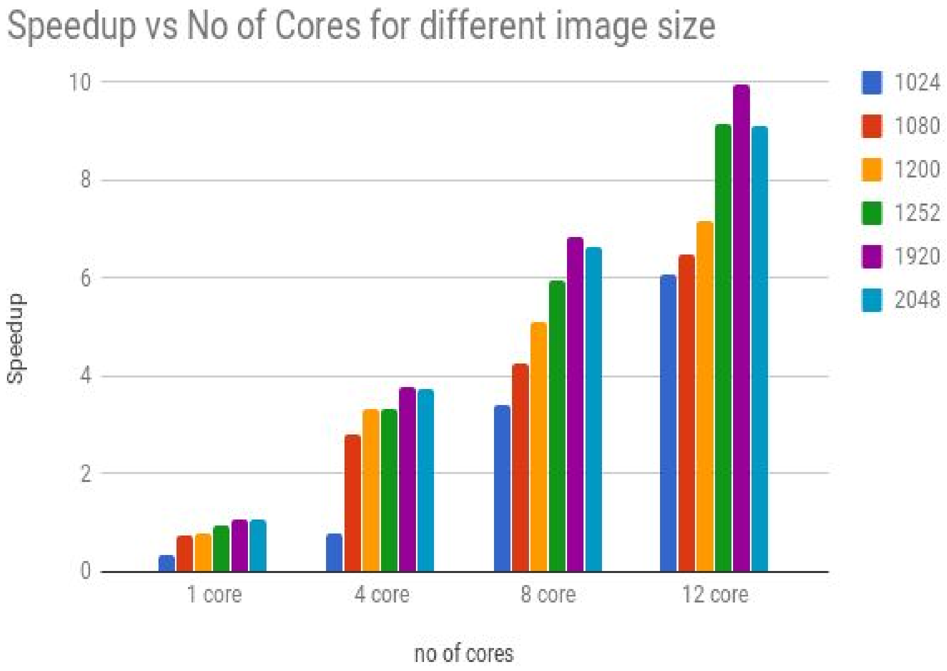


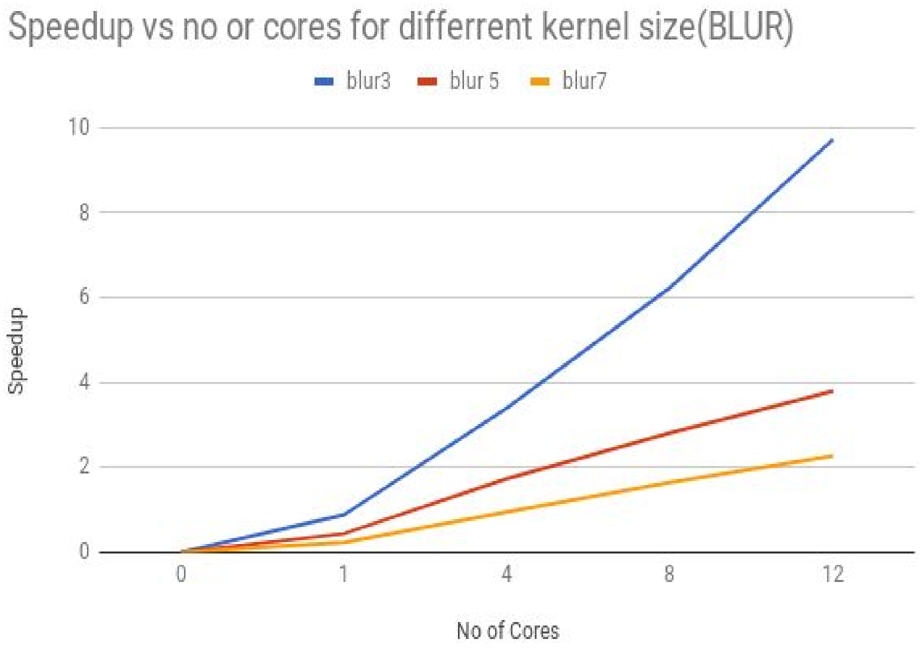
Block wise Parallel Algorithm

- compute the factors of p and store them in an array
- Align the blocks using the factors
- For each pair of factors (a,b) such that ab = p
    - For each pixel present in the present processors block
        - Compute the convolution
        - Store it in the new image

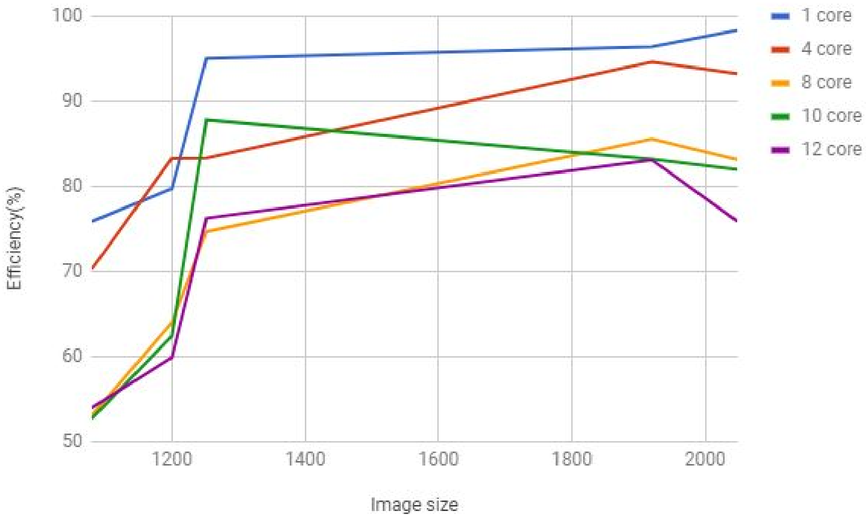
Need: When we parallelize row wise, for a single row we have already accessed its neighbouring rows and we are not reusing them. So temporal locality of thows pixel is not used. To overcome that we divide the work block wise so that temporal locality of the pixels can be used. Here we are using different combinations of blocks to know which one is better for what conditions.

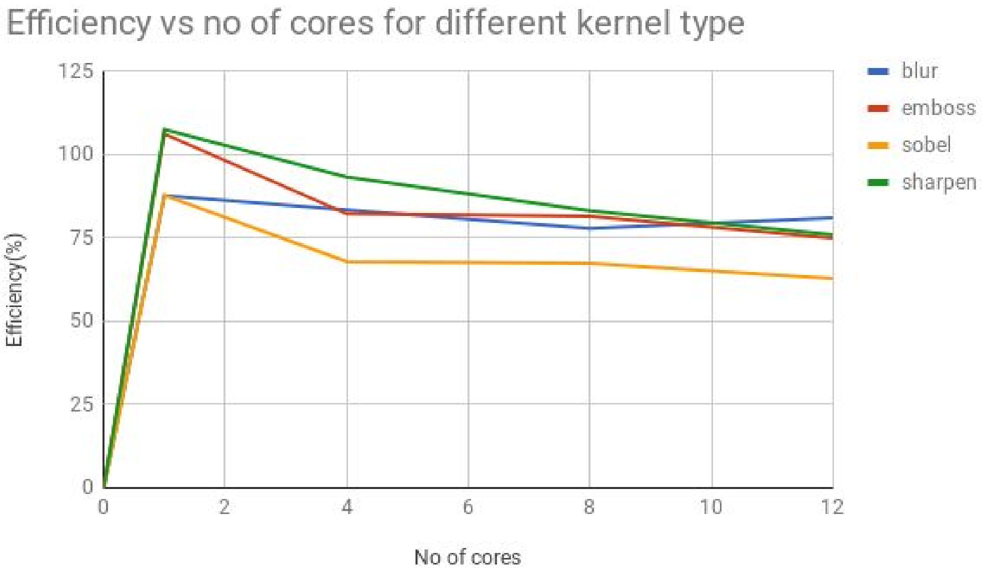
Speedup using Row-wise parallel



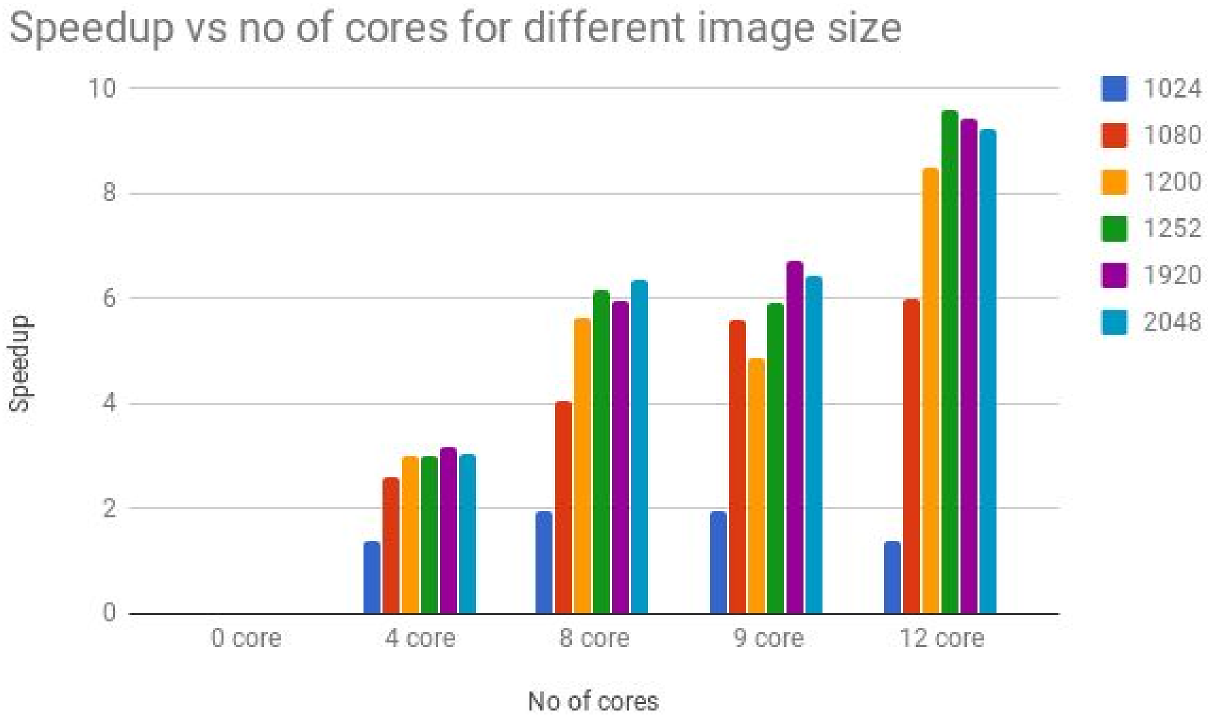


Efficiency for Row-wise parallel

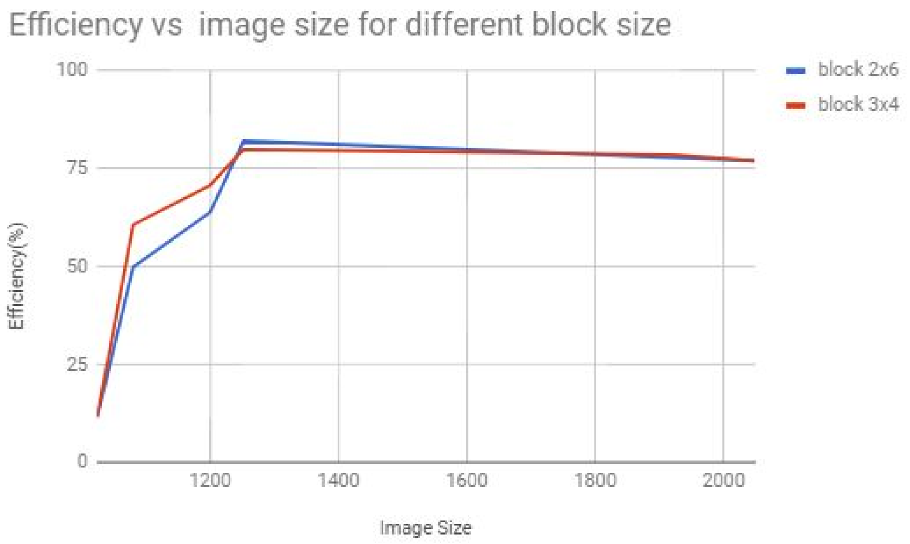


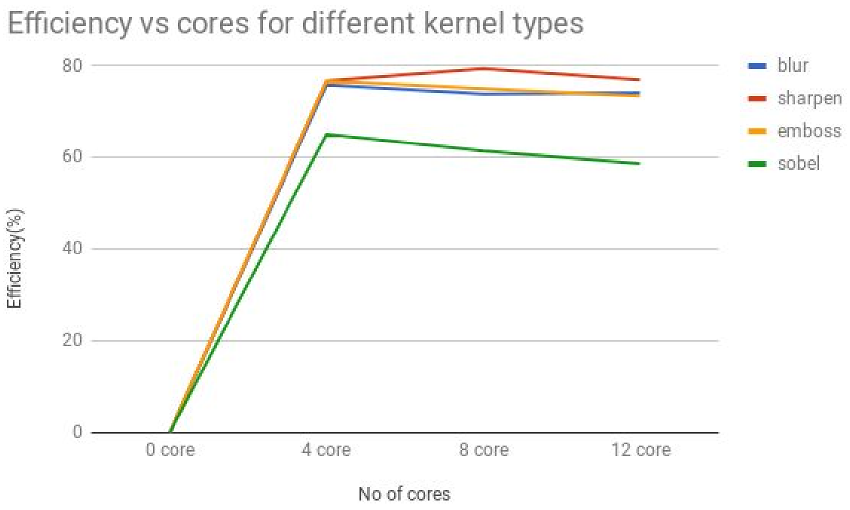


Speedup using block-wise parallel



Efficiency for Block-wise parallel





Performance Analysis

Using Amdahl’s Law:

p = 12 , Image size = 2048x2048, Kernel size = 3x3

s = 18 / (18 + 2048 * 2048 * 3 * 3 * 3)

s = 1.589 * 10-7

Speed up <= 1/(s + (1-s)/p)

Speedup <= 11.99997

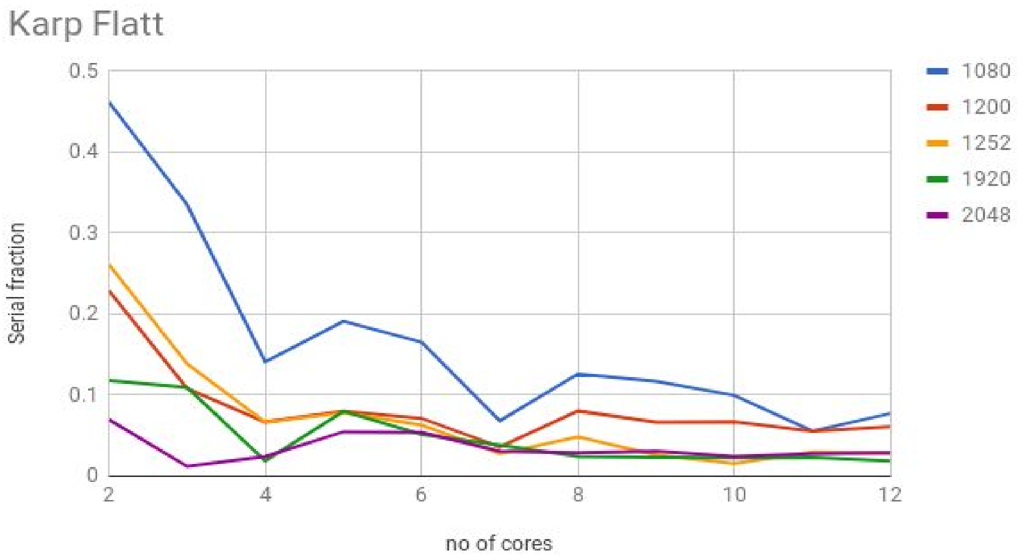
Using Gustafson - Barsis Law

s = 1.589 * 10-7

Speedup <= p + s(1-p)

Speedup <= 11.99999

Karp Flatt Metric analysis



- As the problem size increases, serial fraction decreases. Because the serial fraction doesn't consist of any loop that is dependent on problem size. So Serial computation is constant while parallel computation increases which implies serial fraction decreases.
- As the number of cores increases serial fraction decreases which implies more cores we use we get more and more speedup.