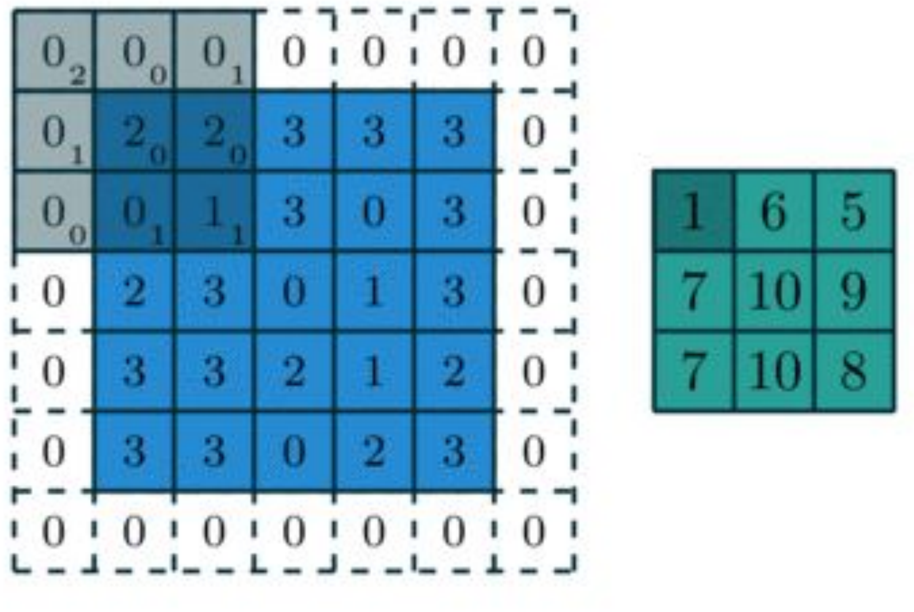
EC500 E1: Parallel Programming and HPC

Parallelizing 2D Convolutions on GPUs

Group 1: Michael Clifford, Patrick Dillon, Frank Tranghese

The 2D Convolution Example



2D Convolutions generate new representations of the original data matrix based on a linear combination of its elements with that of a filter matrix or “Kernel” as it slides through the original data.

One relevant reason for parallelizing and speeding up this operation is its wide use in computer vision and modern deep learning applications.

Image from Professor Lei Tian, BU Dept. ECE

The 1D Convolution Matrix (Circulant Matrix)

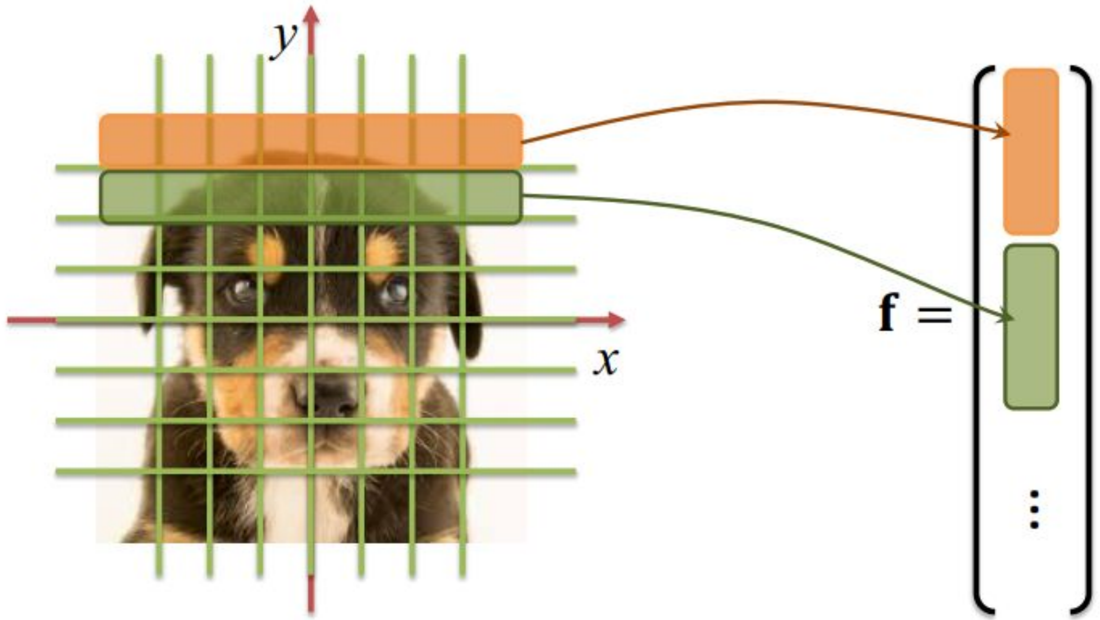
$$\mathbf{g} = \mathbf{A}\mathbf{f} = \mathbf{a} * \mathbf{f}$$

$$\mathbf{g} = \begin{array}{c} \begin{array}{cccccc} & & \text{PSF} & & & \end{array} \\ \begin{bmatrix} a_0 & a_{N-1} & a_{N-2} & \cdots & \cdots & a_1 \\ a_1 & a_0 & a_{N-1} & \ddots & & \vdots \\ a_2 & a_1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & a_{N-1} & a_{N-2} \\ \vdots & & \ddots & a_1 & a_0 & a_{N-1} \\ a_{N-1} & \cdots & \cdots & a_2 & a_1 & a_0 \end{bmatrix} \end{array} \begin{pmatrix} \mathbf{f}_0 \\ \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_{N-1} \end{pmatrix}$$

- Here, **PSF** is the **Point Spread Function**, which is also called the transfer function, kernel, filter, etc.

Image from Professor Lei Tian, BU Dept. ECE

Extending into the 2D Space...

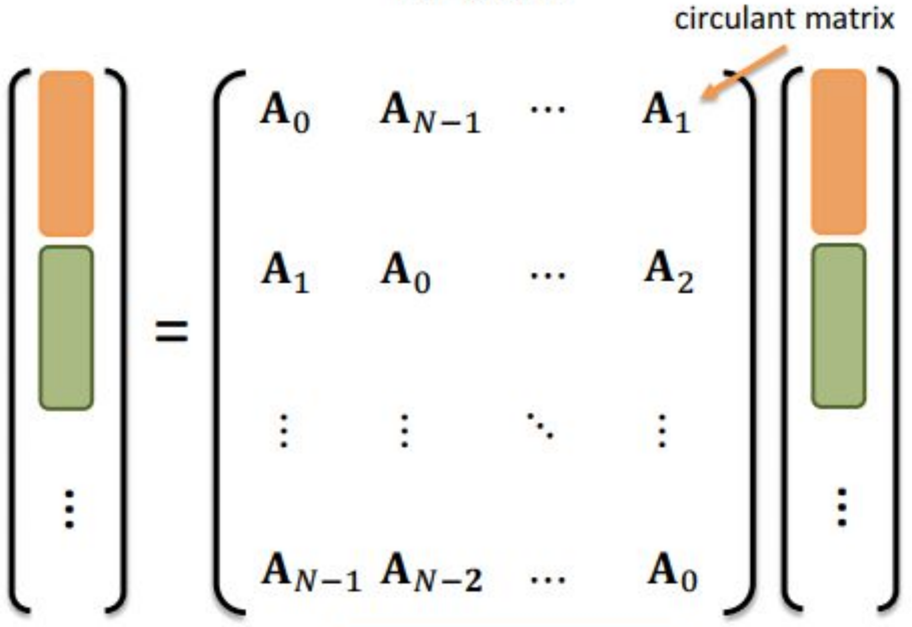


- We can represent the entire image as a column vector of stacked pixel-rows of the image

Image from Professor Lei Tian, BU Dept. ECE

Using Our Stacked Column Representation...

$$\mathbf{g} = \mathbf{A}\mathbf{f} \qquad \mathbf{g}_{mn} = \sum_{m'=0}^{N-1}\sum_{n'=0}^{N-1} \mathbf{a}_{m-m',n-n'}\mathbf{f}_{m',n'}$$


circulant matrix
A_0 A_{N-1} $\cdots$ A_1
A_1 A_0 $\cdots$ A_2
A_{N-1} A_{N-2} $\cdots$ A_0

- Here, we can perform the 2D convolution image in a similar way to the 1D convolution, except here each A_n is itself a circulant matrix.
- If image is NxN, then A will be N^2xN^2 which can get rather large and take up a lot of memory.

Our Image Column Vector

Image from Professor Lei Tian, BU Dept. ECE

Our Plan

- Build the 2D convolution matrix and the resulting matrix in a parallel fashion
  - Building the 2D convolution matrix follows a specific pattern and can be parallelized
  - Reshaping the image, performing matrix math, and reshaping back to original size are all also parallelizable operations.
- Compare to taking the FFT2 of both image and kernel, and taking the IFFT2 of their multiplication
  - FFT2 also parallelizable
  - Test both parallelized methods and compare/contrast for different inputs (different sized images, heat transfer equation, Convolutional Neural Networks?)
- Use OpenACC for GPU parallelization
  - OpenACC looks similar to OpenMP

Coding Plan

- Begin coding of sequential algorithm
    - Creation of circulant matrix and 2D representation
    - Convolution via matrix multiplication
- In parallel, install plumbing…
    - Library for I/O of images as 2D matrices (CImg may be an option)
    - Setup script for OpenACC on SCC (BU Help article)
    - Devise plan for testing against FFT
- Turn sequential algorithm into parallel
    - First step is to parallelize convolution/matrix multiplication
    - May also be able to parallelize creation of kernel matrices

Testing, Results, & Other Ideas

- A simple kernel, blur filter, to average pixel values
- We can subsequently develop a more sophisticated kernels
- *Concrete Mathematics* has other interesting convolution applications (e.g. samplesort)
- The work we are proposing could be the beginning of a convolutional neural network