# Image Zooming and Upscaling Using CUDA

GPU Programming course
A.Y.2022/2023


Politecnico di Torino

Marchetti Laura [s294767]
di Gruttola Giardino Nicola [s298846]
Durando Luca [s303395]

# Contents

# Abstract

In this document we implement an upscaling algorithm in CUDA: It is a technique used to produce an enlarged picture from a given digital image while correcting the visual artifacts originated in the zooming process.

Our zooming algorithm works on RGB images in PPM format: it provides as output a zoomed picture, cut out from the original one passed through the command line, while simultaneously correcting its aliased behavior by implementing a convolution with a specific filter.

The user needs to set, through the command line, the RGB picture that has to be zoomed, the coordinates of the center of the selection zone and the side length of the selection mask which has a square odd shape. It is also left to the user to specify the filter to be applied in the convolution: it's both possible to set out a custom kernel or to create a Gaussian filter specifying its length and $\Sigma$.

# 1 Related works

## 1.1 The Pixel Replication Algorithm

In our implementation of the zooming algorithm we develop the gpu version of the already existing algorithm named "Pixel replication" also known as the "Nearest neighbor interpolation".
As its name suggests, it replicates the neighboring pixels in order to increase them to enlarge the image: it creates new pixels from the already given ones.
In this method each pixel is replicated n times row wise and column wise: therefore the size of the final image corresponds to (. . . )
This Algorithm has the advantage of being a very simple technique of implementing the zooming technique but, on the other hand, as the zooming factor increased the resulting image gets more blurred.

## 1.2 Image Filtering Convolution Algorithm

In order to implement the convolution of the zoomed image, with a specified filter, we had to manipulate the already known version of the "Image filtering through convolution" Algorithm. In the original implementation of the algorithm ..

# 2 Proposed Method

In this section we will give to the lecturer a detailed description of the proposed algorithm: some implemented procedures will be explained through pseudocode. The algorithm works in successive stages as described in the following:

- Kernel loading
- Image reading
- Image cut out
- Image enlargement and filling
- Image enhancement

## 2.1 Kernel Loading

The algorithm can choose between custom kernels given as input from files, or gaussian ones, which are generated on the fly, taking as input the length of the kernel and the $\Sigma$ value. Once generation is done, the kernel is loaded into the constant memory of the GPU.

## 2.2 Image Cut-Out

The aim of this step is to select the part of the original image that has to be trimmed: the dimension of the cut area is passed through command line and the script subsequently calculates the dimension of image in output.
The measure of the side of the final image is computed choosing the largest multiple, smaller than the smallest dimension between width and height of the original image. These checks are performed in order to avoid the creation of a final image with a dimension that is not a multiple of the side of the cut area.
The function carries out the logic explained down below:

$$img_out[tid] = img[starting\_byte + row\_offset + column\_offset] \quad (1)$$

where *tid* is the thread id, *img_out* is the final image, *img* is the original image, *starting_byte* is the starting byte of the cut area, $row\_offset$ is the offset of the row of the pixel to be copied and $column\_offset$ is the offset of the column of the pixel to be copied. The implementation is basic, it is a simple copy of the pixels from the original image to the final one, done in parallel using CUDA threads.

## 2.3 Image Enlargement and Filling

This step is the one that enlarges the image and fills the holes that are created by the zooming process. The process begins by creating as many threads as are the bytes in the final image, and each one of them computes the value of the pixel to be copied from the original image, so that all holes are filled.
The holes are filled with the help of the pixel replication algorithm. It replicates the

neighboring pixels in order to increase them to enlarge the image: it creates new pixels from the already given ones, and it is the most basic technique of implementing the zooming technique, giving as output an image blurred and with lots of artifacts.
In the first version of the algorithm, while filling the canvas with the enlarged image, a black border is left around the image, which will be used to perform the convolution with the filter, which needs a larger image than the one that is actually returned as output, whichever the position from which to start the zooming from. The final version of the algorithm, instead, performs the enlargement, and if the cutout is not at the border, the program will use neighboring pixels to create a slightly bigger image for the convolution. If the cutout is at the border, instead, the back border will be used just for the border part, leaving a small, unnoticeable shade at the last pixels of the image. This final version allows to have an image which is even more faithful to the original one.

## 2.4 Image Enhancement

# 3 Experimental Results

# 4 Conclusions

In this paper we proposed a technique for zooming and enhancing a digital picture in RGB colors by exploiting the gpu parallelism.