CUDA Edge Detection

*FLORESCU Emilia-Christina*

*gr. CEN3.S1A*

**Introduction**

The project implements an alghorithm for edge detection using CUDA. The purpose of edge detection in general is to significantly reduce the amount of data in an image, while preserving the structural properties to be used for further image processing.

**Algorithm overview**

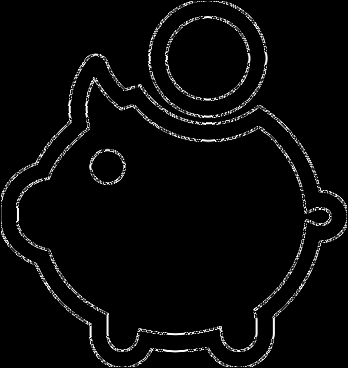
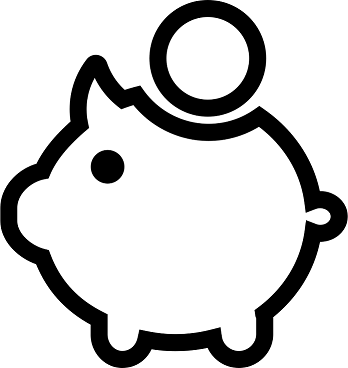
A 3×3 kernel is used, which is convolved with the original image to calculate approximations of the derivatives – one for horizontal changes, and one for vertical. The x-coordinate is defined here as increasing in the "right"-direction, and the y-coordinate is defined as increasing in the "down"-direction.

Next, a pythagorean formula eliminates negative values and generates a medium value based on the X and Y gradients.

At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude, using:

G =

Since the intensity function of a digital image is only known at discrete points, derivatives of this function cannot be defined unless we assume that there is an underlying continuous intensity function which has been sampled at the image points. With some additional assumptions, the derivative of the continuous intensity function can be computed as a function on the sampled intensity function, i.e. the digital image. It turns out that the derivatives at any particular point are functions of the intensity values at virtually all image points. However, approximations of these derivative functions can be defined at lesser or larger degrees of accuracy.



**GPU vs. CPU**

There is a significant difference between the time taken to run the code on the processor, vs. running the same code with CUDA.

For a 150x150px image:

* CUDA: 0.085 ms
* Processor: 4 ms

For a 300x300 image:

* CUDA: 1.92 ms
* Processor: 68 ms

For a 600x600 image:

* CUDA: 2.3 ms
* Processor: 87 ms