INTERNATIONAL JOURNAL OF ENGINEERING EDUCATION (IJEE)
ISSN: 0949-149X May-June2017 Vole 9, No 3
www.sweetmaxwell.org

Analyzing Efficiency of GPU for Executing High Performance Computing Applications

Fatemehossadat Rezvaninejad[1], Fahimeh Yazdanpanah[2]

[1]Bachelor student, Computer department, Faculty of Engineering, Shahid Bahonar university of Kerman

rezvaninezhadf@gmail.com

[2] Assistant Professor of Computer department, Faculty of Engineering, Shahid Bahonar university of Kerman

yazdanpanah@uk.ac.ir

Abstract

Nowadays, given the increased function of computers and the need for humanity to do more work in their work the need for the processors to be implemented in a way that allows the implementation of several software at the same time as the speed to ideal ratio has been created. The need for computer engineers to do graphic work of high computing time such as photoshop, animation and etc. has also been instrumental in understanding and developing the CUDA parallel graphics and graphics processor. Cuda is a parallel programming language for the graphics processor which can be used in programs with high computational levels to write the program in such a way that the GPU and processor are both part of the implementation of the share until its implementation is reduced find out knapsack problems and image processing today are of great importance. The volume of their calculations increases the run time. In this paper we try to explain the role and importance of Cuda programming in the knapsack problem and image processing. The knapsack problem examines how objects with different weights and values are located in a single-hulled backpack, with the backbone of the highest value. However different approaches are provided according to the type of backpack. In the image processing the fading methods of the images have been discussed.

Keywords: GPU, CUDA, Knapsack problem, Binary knapsack, Image processing, Blurring.

1- Introduction

The faster and cheaper the computers and the ability to broadcast images with the communications technology the more people they have access to. Video conferencing has become a dynamic way to do business and home computers can display and manage images well. Fortunately, with the speed of processing and memory of computers in terms of image processing facilities, concerns have been reduced in this case and this trend continues. The science of image processing is one of the most widely used science in engineering and many studies and studies have been done in the past and many improvements have been made. The rapid development of these developments has been so high that now and after a short time the impact of image processing can be clearly seen in many sciences and industries. While some of these uses they are dependent on image processing without it being usable.

Nowadays image processing has become widely used in expanding the various discrete information-processing methods such as scanners and digital cameras. The images obtained from this information always have some noise and in some cases they also have the problem of blurring the boundaries inside the image which reduces image quality. The image processing is called image processing which is used to reduce defects and improve the quality of the image such as rebuilding noisy images, compressing and encrypting images and understanding the image by the machine.

Graphics processors are used on many devices such as smartphones and personal computers. Modern graphics processors are highly effective in controlling the graphics of computers and their balanced structure makes them more efficient than multi-core main processors [1]. In 1999, Nvidia introduced the geforce 256 processor as the world's first graphics processor. The geforce 256 was an integrated single processor processor with render engines that had a processing power of at least 10 million polygons per second. Nvidia's commercial rival, ATI technologies, uses the term visual processing unit (VPU) and shipped the 2700 radeon product in 2002 [1].

The graphics processor includes millions of transistors which is much larger than the transistors in the central processing unit. Today's GPU is used to accelerate applications that are used to process large data. Prior to 1970 the central processing unit was responsible for computing and interpreting images and given the fact that the interpretation of their images required a lot of processing power the speed of the core processor was greatly reduced. After that, graphical processing units were designed and constructed. The number of cores in the GPU is much higher than the number of cores in the core processor which makes it easy to perform calculations on a GPU than the CPU. The control unit and CPU cache are larger than the controller unit and the GPU cache memory. The central processor allocates more transistors to the information processing. The graphics processor is used in parallel programming much more than the central processor and it should be noted that the GPU does not replace the central processor because there are simple algorithms that are more effective on the CPU than the CPU. GPUs are running.

To interface with the graphics processor and coding in it we need to have a soft layer called cuda in parallel. In fact, cuda is the parallel computing architecture of Nvidia. cuda is a parallel programming language with a graphics processor that boosts processing speed and reduces execution time.

The programming process is used to map information elements to parallel processing threads. While the graphics processor has its own advantages, programmers initially have a lot of problems with the transfer of algorithms from the central processing unit to graphics processors were faced. Given that the graphics processor was built for graphic processing and video games, it behaved in accordance with the programming environment and the developer needed a deep understanding of the programming interface and graphics architectural interface, usually these application programming interfaces which is written on the platform.

Cuda is a parallel computing platform and a programming model that was invented by Nvidia. Cuda is a demonstrable demonstration of the implementation of the calculations in order to control the power of the graphical processing unit [2]. Cuda has been developed for the purpose of designing several purposes:

- Provide a small set of extensions of standard programming languages such as C that can implement a direct algorithm. Programmers can focus on parallelization of the algorithm by CUDA C / C over time spent on their execution [2].
- Support for heterogeneous computing: support is available where the applications use the main processor and graphics processor. Here the class part runs on the main processor and the parallel part on the graphics processor. Each has its own storage space. These settings allow simultaneous computing on the original processor and graphics processor without competing in memory resources [2].
- The graphics processor has hundreds of cores that can run thousands of computing threads altogether. These cores have shared resources such as a stable file and a shared memory. The shared memory on the chip allows parallel operations on these cores to share data without sending it through the system memory bus [2].

2- knapsack problem in cuda

The knapsack problem has been studied for more than a century [3]. The knapsack problem is a component of optimization. Suppose we have sets of objects, each of which has a certain weight and value. To each object, numbers are allocated in such a way that the weight of the objects selected is less than or equal to the specified value and their value is maximized. The reason for naming this issue is tourism with a limited size knapsack and should fill it with the most useful of objects. There are usually financial constraints in allocating resources.

How is the knapsack problem? Imagine that tourists want to fill their knapsack by choosing the possible scenarios from the variety of things that it provides the most. Which items are in the top priority for knapsack to be the most significant?

2-1 Dynamic programming

If all the weights ($w_1,w_2,\ldots,w_n,W$) are nonnegative integers the knapsack problem in the case of a polynomial pseudo is capable of solving using dynamic programming [4]. For simplicity assume all weights are positive ($w_i>0$). Here the sum of the value of the selected maximal goods is demanded, assuming that the total weight of them is max. W. Now if for each w <W the value of m [w] is defined as the maximum acquisition value so that the total weight of the objects selected is at most W then obviously m [w] is the desired answer.

m [w] has the following features:

m [0]=0 (Total members of the collection are empty, empty).

$m[w] = \max(v_i + m[w - w_i])$ that v_i is the value of the i-th object.

The maximum value of the empty set is zero. To calculate each of m [w], you must examine the n object. Also the array W, m has an element so the execution time for this algorithm is O (nW). Obviously by dividing w1, w2, ..., W by their largest common parabola, one can optimize the execution time of the algorithm [4]. The convolution of O (nW) is not a contradiction with the NP-COMPLETE knapsack problem since W is not polynomial in contrast to n. The length of the input w is proportional to the number of bits needed to display W, logW, not W.

2-2 Greedy approximation algorithm

The zero and one approach: an object or completely falls into the knapsack or not.

Fragmentation: you can put a part of an object in a collar, for example, you can put two-thirds of an object in a knapsack. Suppose S sets the existing objects. The value of p is the value of an object, w is the weight of the object and W is the weight that the knapsack can tolerate.

Zero and one: the first method is to get all the subsets of S out of the range and then select among the sub-sets that have the highest value and the weight of the objects is not more than the backpack weight. Is this the right way? The answer is definitely not good. The number of subsets of a n-member set is 2n, as it is clear that this path is not suitable for large sets. But what is the solution? one of the ways in previous articles has been to arrange items in order of weight and then put the most in a backpack. This method is also not accepted, because if a device is worth more but its weight is less than the rest or in reverse the result is not necessarily the optimal result of the problem [1].

Another solution that has been proposed is the same way that is, instead of being sorted according to weight we arrange it in value but the result of this method is not optimal, as in the above manner. Because objects that are worth more may have more weight and if placed in a backpack the optimal result is not necessarily achieved. Because the value of several lighter objects may be greater than the value of a large object. So it does not always give an optimal answer [1].

Another solution seeks to solve the failure of the above methods, this way it is said that objects should be in a backpack based on a certain ratio derived from the value to weight ratio. In this way the problems mentioned above are not created. Because there is a ratio, objects are in a backpack in terms of the value to weight ratio so if a value is more valuable and less weight there is a greater chance of being in the backpack [1]. To do this, in the first step the value to weight ratio must be achieved for all things and then arranged them and objects that have a higher proportion are first placed in the backpack until the weight of the objects in the backpack is greater than W do not be.

Fragmentation: in this way, a part of the objects can be placed inside the backpack, for example, two thirds of an object in a backpack [1]. It is like the same way as the value to weight ratio but in the end, if the total value of the objects in the backpack is less than the backpack weight and this weight is less than the weight of an object, you can place a piece of the object in a backpack. This method has a better outcome than previous methods. Is this an example of a breach that is proved to be always not an optimal result?

If there are n objects that have been numbered from 1 to n. The i-th object has a value of v and weighs in w weights and values are generally assumed to be non negative. To simplify the display without diminishing the overall problem, it can be assumed that the objects are arranged in order of magnitude in terms of their weight. The most weight you can carry in a backpack is W [1].

The most common type of this problem is knapsack problem 0 and 1, that is the number of each object is 0 (we do not select that object) or 1 (that object is selected). Knapsack problem 0 and 1 can be expressed in this way in mathematical language: maximize the value of $\sum_{i=1}^{n} v_i x_i$ So that :

$$\sum_{i=1}^{n} w_i x_i \leq W , x_i \in \{0 , 1\} \qquad (1)$$

A knapsack problem with boundary is another version of the question in which the number of objects (xi) is a numerical integer and non-negative integer and is maximally equal to Ci. In mathematical terms, maximize the value of $\sum_{i=1}^{n} v_i x_i$ such that:

$$\sum_{i=1}^{n} w_i x_i \leq W , x_i \in \{0, 1, \dots , C_i\} \qquad (2)$$

A knapsack problem without boundaries does not limit the number of objects that is, from any object to any arbitrary number can be chosen. A version of this most relevant question has the following features: A decision problem is the problem 0 and 1, for each object the weight and value are equal to $w_i = v_i$

In terms of computer science the attention to the knapsack problem has been specially considered because it has an algorithm with a polynomial pseudo time using dynamic programming and also has an approximate polynomial valued algorithm that uses polynomial pseudo time algorithms as a sub program. The exact solution of this question is NP-COMPLETE so there is no prediction that a solution that is simultaneously true and fast (with the polynomial execution time) for each arbitrary input.

2-3 Coverage relations in the knapsack problem without boundary

In solving the knapsack problem without invoking, putting things that are never used can be simplified. For example, suppose for an object such as i one can find a subset of objects named J in such a way that their total value is greater than i and their total weight is less than i so i can not an optimal answer is found. At this time we will cover the collection J. (note that this argument can not be used for the boundary problem of the backbone since it may have already been used and completed from the objects of the constructor J set [1]. Finding covering relationships can help the volume of search space can be reduced to a large extent. There are various types of covering relationships that all apply to the following inequality [1]:

$$x \in Z_+^n \text{ and } \sum_{j \in J} v_j x_j \geq \alpha v_I \text{ for some}, \sum_{j \in J} w_j x_j \leq \alpha w_i \tag{3}$$

that: $J \subseteq N$ ، $\alpha \in Z_+$ ، $i \notin J$,x_j .The number of choices for an object is represented by the type j (note that these js are members of J set.

2-3-1 Optional Coverage

The object i is selectively covered by J, if the sum of the weight of a set of members of J is less than wi and their sum of value is greater than vi, its mathematical expression is this:

$$\alpha = 1 \text{ that } x \in Z_+^n \text{ in case } \sum_{j \in J} v_j x_j \geq v_i , \sum_{j \in J} w_j x_j \leq w_i \tag{4}$$

This kind of overlays is not so easy to calculate in terms of computational computation so the best practices are a dynamic solution. In fact, this is a knapsack problem but with smaller parameters that are as follows:

$V = v_i$, $W = w_i$ and objects bounded by the set J the mathematical symbol of the constitutive constituency is chosen as follows [1].

2-3-2 Extreme Coverage

The object i is partially covered by J If the number of objects of type i by J is covered, it is mathematical expression:

$$\alpha \geq 1 , x \in Z_+^n \text{ in case } \sum_{j \in J} v_j x_j \geq \alpha v_i \text{ and } \sum_{j \in J} w_j x_j \leq \alpha w_i \tag{5}$$

This type of coverage is not a more comprehensive representation of selective overlays and is used in the EDUK algorithm. The smallest α is called the limit i. In mathematical terms:

$$t_i = (\alpha - 1) w_i \tag{6}$$

In this case the maximum optimal response can be included in the number of α-1 objects of type i [1].

2-3-3 Multiple Surfaces

The object i is multiplied by the object j if i is covered by a number of objects of type j. In terms of the following:

$$x_j \in Z_+ \text{ for } v_j x_j \geq v_i , w_j x_j \leq w_i \text{ that } J = \{j\} , \alpha = 1, x_j = \left\lfloor \frac{w_i}{w_j} \right\rfloor \quad (7)$$

This coating can be used to optimize the solution, since the detection of the covering relationships of this kind does not require much computation and is relatively simple [1].

2-3-4 Modular Coverage

Suppose b is the best object, that is, for all i, b. $\frac{v_b}{w_b} \geq \frac{v_i}{w_i}$ is an object with the highest density. The i-th object is covered modularly by object j if it is covered by j and a number of objects of type b (object i is only covered by j and a number of objects of type b and there is no need to use other objects). Math is like this.

$$v_j + tv_b \geq v_I , w_j + tw_b \leq w_I \quad (8)$$

That: $J = \{b, j\}, \alpha = 1, x_b = t, x_j = 1$

The mathematical symbol of modular overlays is as $i \ll= j$ [1].

3- Image processing

The processing of images has a major pocket enhancement of image and machine vision. Image enhancement involves techniques such as using a fader filter and increasing contrast to enhance the visual quality of images and ensure that they are displayed correctly in the destination environment (such as a printer or computer monitor). While the machine vision is in ways that can help them understand the meaning and content of the images to be used in robotic tasks.

3-1 Computer vision

Computer vision is one of the most modern and diverse branches of artificial intelligence that combines computer image processing techniques and machine learning tools with the ability to visualize objects, landscapes and intelligent understanding of their various features. Visual AI applications, machine learning, machine exploration in information, machine exploration in texts soft computing, fuzzy logic, image processing.

3-1-1 Main tasks in computer vision:

1. Object detection: detect the presence or mode of an object in an image, for example:

- Search for digital images based on their content (image centered retrieval)
- Identify human faces and their position in photos
- Estimation of the 3D state of humans and their organs

2. Tracking: tracking known objects among a number of images straight away. For example, following a person when he walkes into a shopping mall.

3. Landscape interpretation: making a model of ani. For example, building a model from the periphery with the help of images taken from the camera mounted on a robot.

4. Auto-location: specify the location and movement of the camera as a computer vision member such as routing a robot in the museum.

3-1-2 Computer vision systems:

A computer system can be divided into the following subfields:

1. Imaging: an image or sequence of images taken with a shooting system (camera, radar, tomography system). Usually the imaging system should be set before use.

2. Preview: the image is exposed to the "low level" function in its preprocessing step. The goal of this step is to reduce noise and reduce the overall amount of information. This is accomplished by employing a variety of digital image processing techniques such as image sampling, digital filters, gradient x and y calculations and possibly gradients, image segmentation, pixel threshold calculation, Fourier transform, estimation of movement for local areas the image, also known as light flow estimation, estimates the discrepancy in highlighting and multi dimensional images.

3- Feature extraction: the purpose of the feature extraction is to turn raw information into a better form for subsequent statistical processing. In fact, in order to distinguish from the patterns of an image or the creator of that image there must be a number of generic or specific features of the image that they call extraction of the feature. For example, in identifying signatures by image processing, a number of features (such as line gradients) are extracted from the scanned image by which the signature can be identified. border detection extraction of corner features, extracting images The rotation of deep maps the acquisition of alignment lines and possibly the passage of bending zeros are part of the use of feature extraction.

3-2 Image processing

The distinct boundary between image processing on the one hand and the machine vision can not be determined on the other hand However it can detect three types of low level, mid level and high level processing.

- Low level processing includes basic processes such as preprocessing to remove Noise, improve contrast and image filtering. The characteristic of this kind of processing is that the input and output are image.
- Intermediate processing involves segmentation of the image in order to divide it into different areas and objects, describing objects in a way that is suitable for computer processing and categorizing or recognizing various objects. The characteristic of this process is that its input is usually the image and output of attributes of image objects such as edges, contours and objects.
- High level processing involves understanding the relationships between detected objects the interpretation of the scene and the interpretation and diagnosis that the human eye system does. Many high level processes are in the field of machine vision.

In fact, image processing is a kind of signal processing the input of which is an image and the output can be an image or anything that is related to the image. Changing the color of a color to gray image is an example of image processing and fingerprint recognition is another example of image processing applications. Most image processing is applied to the entire image and similar steps apply continuously to all image pixels. An image can be represented by a two dimensional function f (x, y) where y and x are the spatial coordinates and the value of f at each point. The intensity of the image illumination in That point is called. The term gray level also refers to the high brightness of monochrome images. Color images also consist of a number of two-dimensional images [1]. When the values of x and y and the value of f (x, y) are expressed in terms of discrete and finite values they call the image a digital image. Digitizing x and y values are gradual sampling and digitization of the value of f (x, y). A digital image is composed of a number of finite elements with different amounts and positions. These elements are called pixels. Digital image processing involves applying various processes to a digital image using a digital computer.

3-2-1 image processing on serial and parallel processors

In this section we discuss the role of the serial processor and parallel processor in image processing. Image processing to run on a GPU is ideally suited for any pixel can directly be placed on a separate thread. In processing the image of the graphics processor and the main processor play separate roles in the following order:

Main processor: taking pictures (camera), graphics processor: processing, main processor: operations related to processing results, main processor: erase memory

3-3 Fade Box

The dark box, also known as the linear filter box is a spatial linear domain filter whose each pixel in the output image has a value equal to the average values of its neighbors in the input image. This filter is a low pass filter. A box of 3 to 3 can be written as the 9.1 matrix [15].

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \tag{9}$$

Given the use of equal weights, this can be done using much simpler algorithms. [15].

Code fade: In the main.cpp file, we will load and save the images and call the filter:

```
int main(int argc, char** argv) {

if(argc != 3) {

std::cout << "Run with input and output image filenames." << std::endl;

return 0;

}

const char* input_file = argv[1];

const char* output_file = argv;
```

```
std::vector<unsigned char> in_image;

unsigned int width, height;

unsigned error = lodepng::decode(in_image, width, height, input_file);

if(error) std::cout << "decoder error " << error << ": " << lodepng_error_text(error) << std::endl;

unsigned char* input_image = new unsigned char[(in_image.size()*3)/4];

unsigned char* output_image = new unsigned char[(in_image.size()*3)/4];

int where = 0;

for(int i = 0; i < in_image.size(); ++i) {

if((i+1) % 4 != 0) {

input_image[where] = in_image.at(i);
```

```
    output_image[where] = 255;
    where++;    }   }
    filter(input_image, output_image, width, height);//  filtering
    std::vector<unsigned char> out_image;
    for(int i = 0; i < in_image.size(); ++i) {
    out_image.push_back(output_image[i]);
    if((i+1) % 3 == 0) {
    out_image.push_back(255);     }    }
    error = lodepng::encode(output_file, out_image, width, height);
    if(error) std::cout << "encoder error " << error << ": "<< lodepng_error_text(error) << std::endl;
    delete[] input_image;
    delete[] output_image;
    return 0; }[17]
```

This function converts the image into a vector then we filter the image information into the filter function (the filter function loads the information in the graphics processor and invokes the cuda core function that executes the filter). We will ultimately store and remove the image [17].

```
    void filter (unsigned char* input_image, unsigned char* output_image, int width, int height) {
    unsigned char* dev_input;
    unsigned char* dev_output;
    getError(cudaMalloc( (void**) &dev_input, width*height*3*sizeof(unsigned char)));
    getError(cudaMemcpy( dev_input, input_image, width*height*3*sizeof(unsigned char),
cudaMemcpyHostToDevice ));
    getError(cudaMalloc( (void**) &dev_output, width*height*3*sizeof(unsigned char)));
    dim3 blockDims(512,1,1);
    dim3 gridDims((unsigned int) ceil((double)(width*height*3/blockDims.x)), 1, 1 );
    filter<<<gridDims, blockDims>>>(dev_input, dev_output, width, height);
    getError(cudaMemcpy(output_image, dev_output, width*height*3*sizeof(unsigned char),
cudaMemcpyDeviceToHost ));
    getError(cudaFree(dev_input));
    getError(cudaFree(dev_output)); }
```

In the above function we copy and paste the data into a graphic processor. The information is in the graphics processor to allow readability by the kernel. Memory is allocated to the output image and to the filter result position. This function is located in kernels.cu [12]. In this part of the program for each pixel the input image gains the average pixels of all its neighbors and we write in the output image.

```
__global__void blur(unsigned char* input_image, unsigned char* output_image, int width, int height) {
    const unsigned int offset = blockIdx.x*blockDim.x + threadIdx.x;
    int x = offset % width;
    int y = (offset-x)/width;
    int fsize = 5; // Filter size
    if(offset < width*height) {
```

```
float output_red = 0;
float output_green = 0;
float output_blue = 0;
int hits = 0;
for(int ox = -fsize; ox < fsize+1; ++ox) {
for(int oy = -fsize; oy < fsize+1; ++oy) {
if((x+ox) > -1 && (x+ox) < width && (y+oy) > -1 && (y+oy) < height) {
const int currentoffset = (offset+ox+oy*width)*3;
output_red += input_image[currentoffset];
output_green += input_image[currentoffset+1];
output_blue += input_image[currentoffset+2];
hits++;         }       }     }
output_image[offset*3] = output_red/hits;
output_image[offset*3+1] = output_green/hits;
output_image[offset*3+2] = output_blue/hits;} }
```

4 - Gaussian Blur

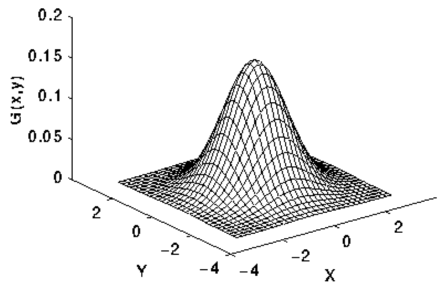
Image smoothing is a type of convolution that is often used to reduce interference and image details and this process is often done by a low pass filter. The filter stores low frequency frequencies while decreasing high frequency values. In fact the image is smooth by reducing the imbalance between pixels [12]. Image smoothing is sometimes used as a preprocessor for other operations on the image and most commonly, Image smoothing is used to reduce interference before applying a border detection algorithm. smoothing can be applied repeatedly to an image in order to obtain the desired effect [12].

An easy way to achieve smoothing is to use the mean filter. This idea is expressed by replacing each pixel with the mean of all neighboring pixels. One of the benefits of this is its simplicity and speed. However the main drawback of this method is the remote locations, especially those that have the furthest distance from the pixel and can give a false view of the average neighbors' ture [13]. Another way of smoothing is an image using gaussian blur. This method is a high level smoothing of images because it reduces the amount of high frequencies corresponding to its frequency and gives less weight to pixels away from the center of the window [13]. The gaussian function is defined as:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (10)$$

In this equation the parameters are as follows:

σ: the fading factor increases if the image is smoothing. E: Euler number, x: Horizontal distance from center pixel, y: Vertical distance from center pixel given this equation the x and y distances for the central pixel will be zero. While the distance from the center pixel increases by $x^2 + y^2$ and the weight decreases [7].



(Fig. 1): the discrete nucleus in (0,0) and 1 = σ

4-1 Identify Sobel Edge Detection

Edge detection is a common method of image processing that is used to detect and extract features. Edge detection in an image can significantly reduce the amount of information needed for processing in the next step, while maintaining the original structure of the image. Not be An idea in this regard is to remove everything from the image except the pixels that are part of the edge. These edges have special properties such as corners, lines, curves and so on. A set of these properties or attributes can be used to achieve a larger image in image smoothing. The edge can be detected by local variation of the intensity of the image. The edge usually divides the image into two different areas. Most edge detection algorithms work best on the image on which the interception method is applied. Today's practices use differential operators and high-pass filters [12]. A simple edge detection algorithm adopts the Sobel edge detection algorithm which involves the convolving the image using the correct value filter which is both simple and inexpensive in calculations. The Sobel filter is defined as:

$$S1 = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, S2 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \qquad (11)$$

To apply the Sobel algorithm on the image, first we find the approximate derivatives by keeping the horizontal and vertical directions. If we assume that A is the original image, G_x is the derivative of the derivative in the horizontal axis and G_y is the derivative approximation in the vertical axis:

$$G_y = S_1.B\,,\ G_x = S_1.A \qquad (12)$$

The resulting gradient image is a combination of G_x and G_y. Each pixel G (x, y) of the resulting image can be computed by considering the size of Gx and Gy and the gradients of the direction are calculated as follows:

$$\theta = \tan^{-1}\frac{G_y}{G_x}\,,\ G(x,y) = \sqrt{{G_x}^2 + {G_y}^2} \qquad (13)$$

Finally, to determine whether a pixel of the original image A is part of the edge, we follow the following process: if G (x, y) is bigger than the threshold value then A (x, y) is a part of the edge [12].

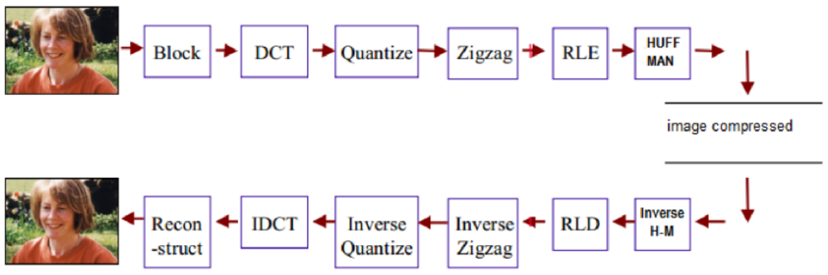
4-2 JPEG compression

as I said, one of the applications of image processing is compression of images. One of the ways to make it in images is JPEG compression developed by a group of photographic specialists to compress non-graphic images (in particular photographs). This group was organized in 1986 and published the JPEG standard in 1992 and In 1994, it was approved as a standard in ISO [16]. This algorithm is based on the fact that the human eye does not see colors at high frequencies [13]. The importance of this algorithm is that the most important compression standard for motion pictures, MPEG, is in fact nothing other than the coding of successive JPEG frames which is an algorithm of various types of compression techniques and consequently the image after a single decryption and encryption step with a slight original image The difference is based on the rate of compression. The JPEG method works based on the DCT conversion and also uses variable length encryption methods such as huffnan encryption.

4-2-1 compression process

The steps for converting a raw image into a JPEG image are:

block preparation, discrete cosine transform, quantization, zigzag scan, run-length encoding, differential scaling, statistical coding.



(Fig. 2): image compression and decompression steps [15]

5. Conclusion

In order to speed up the implementation the GPU was introduced, followed by the parallel programming of the Cuda. This paper attempts to introduce the knapsack problem the ways presented for it the code for its program in cuda, image processing, image cropping, cuda programming, fading the box, compressing images and the importance of programming cuda is in the presentation of this content.

Refrences:

[1]: Taziki, M., Javadian. (1391), Provide a meta-innovative algorithm for solving a two-dimensional backbone with rectangularces.

[2] Amaral, A., Letchford, A.N. (2001). An improved upper bound for the two-dimensional non-guillotine cutting problem, Technical report, Lancaster University, UK.

[3]Arenales, M., Morabito, R. (1995). An and/or-graph approach to the solution of two dimensional guillotine cutting problems, European Journal of Operational Research 84, 599–617.

[4]Beasley, J.E. (1985). Algorithms for unconstrained two-dimensional guillotine cutting, Journal of the Operational Research Society 36, 297–306.

[5] Beasley, J.E. (1990). Or-library: Distributing test problems by electronic mail, Journal of the Operational Research Society.

[6]B.Kirk, D. (2006-2008). NVIDIA CUDA Software and GPU Parallel Computing Architecture. CUDA C PROGRAMMING GUIDE. (2012). www.nvidia.com.

[7]Bozkurt, F., Yağanoğlu, M., Baturalp Günay, F. (2015). Effective Gaussian Blurring Process on Graphics Processing Unit with CUDA.

[8] Pisinger, D. (2003). Where are the hard knapsack problems?. Department of Computer Science, University of Copenhagen, Copenhagen, Denmark.

[9] Mathews ,G. B., (25 June 1989) "On the partition of numbers", Proceedings of the London Mathematical Society 28, 486–490.

[10]G.Hammer, Gallo;P.L, Simeone. (1980). "Quadratic knapsack problems", Mathematical Programming Studies 12: 132–149.

[11]GIOT, Ir.Rudi., Abilio RODRIGUES E SOUSA, Ing. (2012). Image processing algorithm withCUDA for Pure Data. CUDA C PROGRAMMING GUID, www.nvidia.com. Cook, ShaneCUDA PROGRAMMING. (2013), 13-45.

[12]Tse, J. (2012). IMAGE PROCESSING WITH CUDA, 30-34.

[13]Marcus, M. (2014). JPEG Image Compression.

[14]Shaban AL-Ani, M., Hammadi Awad, F. (2013). THE JPEG IMAGE COMPRESSION ALGORITHM, International Journal of Advances in Engineering & Technology.

[15]Jarosz, W. (2001), Fast Image Convolutions.

[16] https://jpeg.org/jpeg/index.html.

[17] http://madsravn.dk/posts/simple-image-processing-with-cuda. www.stackoverflow. com. https://github.com.