CATEGORY: VIRTUAL REALITY & AUGMENTED REALITY - VRAR02

POSTER P6168

CONTACT NAME Bruno Augusto Dorta Marques: brunodortamarques@gmail.com

GPU TECHNOLOGY CONFERENCE

GPU Accelerated Method for Estimation of Light-Sources

Bruno Augusto Dorta Marques Esteban Walter Gonzalez Clua

Instituto da Computação, Universidade Federal Fluminense - RJ - Brasil

Abstract

A real-world lighting environment is composed of a complex combination of physics-based interactions which are difficult to simulate and predict. Several methods to simulate these interactions have been proposed to overcome this difficulty, such as raytracing and global illumination algorithms. However such techniques that are able to recover an entire real-world scene lighting configuration and properties have not been invented yet.

The estimation of a light source configuration of a real-world environment can benefit a wide range of applications. Intelligent applications can produce different behaviours based on the lighting that is present in the user environment. For example, adjusting the color scheme of the interface or changing the brightness and controlling the white-balance of a display.

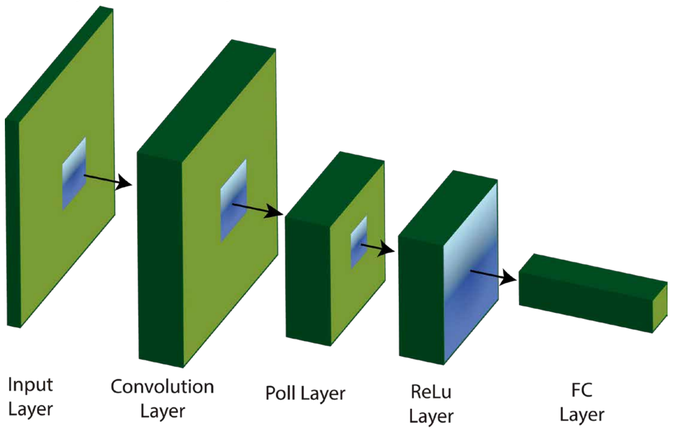
Some particular interesting areas in this scope are the augmented reality and mixed reality. This kind of application requires that both real world and virtual elements have a consistent appearance and lighting conditions. This study propose a GPU accelerated method capable to recognize some aspects of the light source of a real environment. The method is built with a fast evaluation function that fits the highly constrained time of a real-time application.

Introduction

Convolutional neural network is a machine learning approach that has been used to solve a large number of problems including image classification, localization and segmentation problems. A Convolutional neural network usually requires a huge amount of data to be computed and takes the advantage of parallel computing techniques to accelerate the training and evaluation process. These characteristics make the Convolutional neural network algorithms a suitable problem to be solved with a GPU computing technology, such as Nvidia CUDA.

A Convolutional neural network is composed of different types of layers, each layer performs some operations over the output of a previous layer. There are five basic layer types:

- Input Layer - The input layer holds the input data, represented as a 3D volume.
- Convolution Layer - Computes the output of neurons connected to a region in the input volume. The output is computed based on a convolution operation.
- Relu Layer - Computes a elementwise operation over the input.
- Pool Layer - Reduces the input 3D volume size.
- Fully Connected Layer - Computes the output score for each class.



Method

The database of the method consist of a set of images labelled with light-source properties such as light direction, light color, light distance. For the input in the neural network we choose to perform a segmentation of the images to isolate a single object (an human arm) from the real scene.

The Segmentation step is performed on the GPU with a CUDA implementation. The method of segmentation uses the results of Gomez et al. (Gomez,2002), where a simple rule is obtained for the segmentation of the skin. We use the same colorspace found in one of the rules of Gomez et. al method. We adjust the threshold of the rules to perform a cleaner segmentation of our database. The input (r,g,b) is in the format of a normalized RGB. The normalization and the final adjusted rule follows:

$$r = \frac{r}{(r+g+b)},$$

$$g = \frac{g}{(r+g+b)},$$

$$b = \frac{b}{(r+g+b)}$$

$$\frac{r}{g} > 0.9$$

$$\frac{r*b}{(r+g+b)^2} > 0.095$$

$$\frac{r*g}{(r+g+b)^2} > 0.102$$

The total memory size of the database was of about 8 GB, each segmented image contains only one object (an human arm) from the real scene. The CUDA implementations take the input database and split it in several batches of files, each batch is processed in the GPU, an additional step is performed where all the high resolution images are resized to a size of 256*256 pixels.

The Convolutional neural network that we use in this work is the googLeNet. The network creation, training and tests were performed in the GPU, through the Nvidia DIGITS2 and Caffe deep learning frameworks.

The tests for the segmentation and the deep learning process were performed using computers whose specifications are listed below, respectively:

Intel i7-3820 @ 3.60 Ghz, 32GB RAM, Geforce GTX 680, Tesla K20c
Intel i7-3820 @ 3.60 Ghz, 32GB RAM, 2 x Geforce Titan X

And overview of the method is described below :

1. **Group the input images in sets of fixed pixel size.**
2. **For each set of images, perform a parallel segmentation based on the pixel color of the image.**
3. **Resize each image to a fixed size of 256 * 256 pixels.**
4. **Feed the neural network with the database of post processed images.**
5. **Perform a parallel feed forward operation of the first layer of neural network.**
6. **Take the output of the first layer and feed forward to the next layer.**
7. **Repeat step 6 until it reaches the final Fully Connected Layer.**
8. **Perform a back propagation - stochastic descent gradient from the last layer to the previous one.**
9. **Repeat the step 8 until it reaches the first layer.**
10. **Calculate the accuracy and loss, update the weights, repeat the process from step 5 for N epochs.**

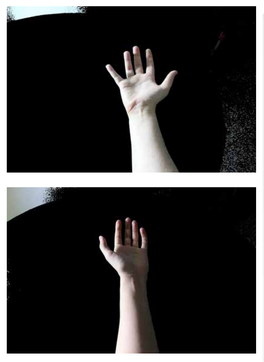
Results

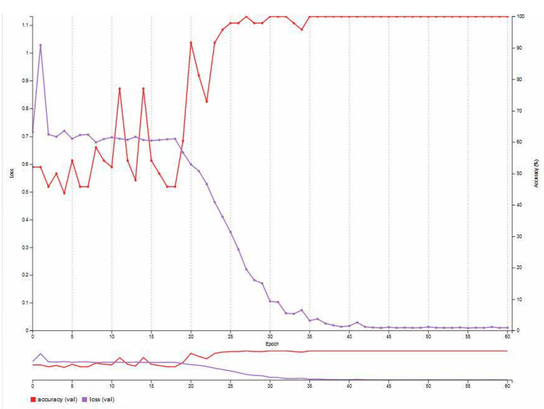
When comparing the GPU accelerated segmentation process with the CPU single core version, we have reached a speedup of about 482%, as expressed in the table below:

Size of the database (MB)	CPU Total Time (ms)	GPU Total Time (ms)	GPU Kernel + MemCopy Time (ms)	Speedup
1025.16 MB	4087 ms	851 ms	204 ms	480%
2050.31 MB	7978 ms	1673 ms	408 ms	476%
4100.62 MB	16250 ms	3321 ms	816 ms	489%

The GPU version of our segmentation algorithm performs it faster than the sequential CPU version. However, the bottleneck of the algorithm is the memory management, which 90.02% of the execution time it was being used for the memcpy operation. The CUDA copy engine allows the algorithm to perform it faster, since there is an overlap between the two-way copy operations and the kernel execution.

For the Deep Learning approach, we achieved an accuracy of 100% and a loss value of 0.017 over our small database at the 40th epoch of the training. The Deep Learning training process took less than one minute to finish and the evaluation time is suitable for a Real Time Application.





Conclusions and Future Work

The GPU accelerated platform was verified to be optimal for the proposed application. The well developed GPU algorithms for deep learning development provided by the Nvidia DIGITS and Caffe framework allowed the project to increase the database size and the complexity of the neural network without much effort. Further improvements in the segmentation process needs to be investigated, for both performance and quality of the segmentation. Its is important to note that the method proposed scales well with the size of the database.

Further work and tests are planned in order to evaluate our results under different kinds of environment and different input databases. In future works we plan to expand the database, obtaining new images for the process and perform additional research on different convolutional neural network architectures.

Bibliography

Gomez, Giovani, and E. Morales. "Automatic feature construction and a simple rule induction algorithm for skin detection." Proc. of the ICML workshop on Machine Learning in Computer Vision. 2002.