Wilson Pena

05/09/2022

Final report

Huffman Coding for Image Compression

Final Report

**Introduction**

The purpose of this project was to pick a regular software application and implement it using CUDA. In my project I picked an image compression algorithm named Huffman Coding. The Huffman coding algorithm is a lossless data compression technique which encode/decodes input characters. The lengths of the code given to the inputs characters is determined by the frequency of the related character. The smallest code is assigned to the most frequently occurring character, whereas e largest code is assigned to the least frequently occurring character.

**Implementation**

This program uses multiple files to run. The list of files is listed below with their corresponding utilization. In order to run this program type in the terminal.

>> make; ./compression

* **Compression.cu**
  + This is the main file that drive and execute the whole program.
* **Compress\_Helper.h**
  + This files contains all the functions needed to implement the Huffman Code using regular C++.
* **Compress\_Helper\_cu.h**
  + This files contains all the functions needed to implement the Huffman Code using CUDA in C++.
* **Utilities.h**
  + This files contains multiples functions for the devices and the host, which helps with the execution of the program.
* **MakeFile**
  + It builds and compile the program. Also, it creates an executable.
* **Lena.bmp**
  + This is the image used for compression.

Functions inside **Compress\_Helper.h**/**Compress\_Helper\_cu.h**

* readBMPFILE
  + import and load image.
* occurrence
  + Find probability of occurrence.
* nonZero\_ocurrence
  + Find number of non-zero occurrence.
* minProp
  + Calculate minimum probability.
* sortHist
  + Sort histogram.
* BuildTree
  + Build the Huffman Tree.
* AssignCode
  + Assign Huffman Code to each pixel.
* PrintHuffmanCode
  + Output the Huffman Code for each Pixel.
* calBitLength
  + Calculate and output the average amount of bits used to encode the image.

C++ CUDA

Just C++

Main

calBitLength

PrintHuffmanCode

readBMPFile\_cu

Occurrence\_cu

nonZero\_ocurrence\_cu

minProp\_cu

sortHist\_cu

BuildTree\_cu

AssignCode\_cu

AssignCode

sortHist

minProp

nonZero\_ocurrence

ocurrence

BuildTree

readBMPFile