**Project: Image Compression Using Huffman Coding**

Name: My Name Here

Class and Semester: CSPB2270 , Fall

Images are one of the widely used data sources for robotics and machine learning. Any machine learning algorithm require the captured image by the image sensor to be transferred to the destination computer for processing but if the image size is large this data transfer could take significant amount of time and due to bounded bandwidth of communication lines can limit the frame rate of the camera. In another perspective, large image data can take significant amount of hard disk memory if they are being logged. An algorithm which can compress images without loosing their information content seems necessary to remedy problems mentioned above.

As a homework we discussed a method for compressing text in a lossless way using Huffman Encoding technique which seems to be applicable to the image data as well. A monochrome image consists of a n\*m matrix of char variables where each index corresponds to each pixel of image. By comparing image pixel values to their neighboring pixel, one would realize that if there are many pixels where values of the pixels are repeated as long as those pixels have same color. Repeated pixel values are in image are similar to repeated alphabet letters in written language as we used in the homework where more repeated pixel values could use less number of bits to represent them and this would potentially result in a compression which would take less space to store the image or transfer it through network.

I would be creating a CompressedImage Class which I would suggest to have the following member functions and variables.

**Data Structs**

* Image struct
  + This struct would store the non-compressed image
* CImage struct
  + This would stored the compressed image result.
* Freq\_info
  + Similar to the one in homework it’s a node of Huffman codec tree

**Publics Member Functions for user to interact with:**

* bool LoadImage(string file\_name)
  + this function would take an image file name in .BMP format including its relative path to executable and load it to
  + using the file path it would read the image and store in private member variable (MyImage)
  + if reading was successful it would return true, otherwise false
* bool CompressTheImage()
  + this would call my private member functions to compress the image.(Build Tree,BuildEncodingTable)
  + compressed image will be stored in private member myCImage
  + If compression was successful it would return true, otherwise false
* Shared\_ptr<CImage> GetCompressedImage()
  + This will return a pointer to the compressed image, if images hasn’t been compressed yet it will return NULL

**Private Member Functions Hidden From User:**

* shared\_ptr<CImage> BuildTree()
  + This function would use the private member variable of type Image which stores the input image and create a Huffman codec tree
* void BuildEncodingTable(shared\_ptr<freq\_info> root)
  + it first creates a lookup table for the encoding process. It mapps each symbol in the Huffman codec tree to a string

In order to test this algorithm I’m planning to run it through An example and show how the algorithm works.

**Testing Criteria:**

* I will supply three images with the project and an application.cpp which reads these images and compresses each of them. I will print number of bits used before and after the compression to show the effectiveness of the compression algorithm
* Next I will decompress (by decoding) the compressed image back to its original form and store it in a third matrix and inside two nested loops I would compare every resulted pixel value in the third variable with initial pixel value and if all comparisons pass then I would print success and would print failed otherwise.

References:

[1] Zybook chapter 15

[2] <https://en.wikipedia.org/wiki/Huffman_coding>

[3] <https://www.geeksforgeeks.org/image-compression-using-huffman-coding/> (I learned mostly from this resource but I’m not going to use any of their code in there)