**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

****



**BÁO CÁO ĐỒ ÁN**

**MÔN CS232 – TÍNH TOÁN ĐA PHƯƠNG TIỆN**

**Đề tài: NÉN ẢNH THEO CHUẨN JPEG**

**(THUẬT TOÁN DCT VÀ THUẬT TOÁN HUFFMAN)**

**Giảng viên hướng dẫn: Thầy Đỗ Văn Tiến**

**Chi tiết thành viên nhóm:**

**Lê Phước Trung - MSSV: 20522069**

**Trương Đăng Nghĩa – MSSV: 20521658**

**TP. HỒ CHÍ MINH, Tháng 6 Năm 2022**

**NHẬN XÉT CỦA GIẢNG VIÊN**

TPHCM, Ngày … Tháng … Năm …

Người nhận xét

(Ký tên)

MỤC LỤC

PHẦN 1: MỞ ĐẦU

1. Lý do chọn đề tài………………………………………………………………………………………….4
2. Đối tượng nghiên cứu…………………………………………………………………………………..4
3. Cơ cấu đồ án…………..…………………………………………………………………………………….4

PHẦN 2: NÉN ẢNH THEO CHUẨN JPEG

1. Chuẩn bị ảnh và Chrominance Conversion
2. Chuẩn bị ảnh…………………………………………………………………………………….....5
3. Chrominance Conversion……………………………………………………………………..6
4. Thuật toán DCT và Quantization
5. Thuật toán DCT…………………………………………………………………………………....7
6. Quantization………………………………………………………………………………………..9
7. Thuật toán RLE và Huffman:
8. Thuật toán RLE…………………………………………………………………………………..11
9. Thuật toán Huffman…………………………………………………………………………...12

PHẦN 3: KẾT QUẢ VÀ ĐÁNH GIÁ………………………………………………………………………15

Phần 1: Mở đầu

**1. Lý do chọn đề tài**

Ngày nay với sự phát triển của công nghệ và sự phụ thuộc vào máy tính ngày càng lớn, thì việc chúng ta tìm ra một phương pháp hiệu quả để có thể lưu trữ một lượng lớn data. Ví dụ như một người sở hữu một trang web có sử dụng hang ngàn tấm ảnh, chắc chắn sẽ sử dụng một vài cách để có thể nén ảnh lại. Vì để lưu trữ số lượng lớn tấm ảnh chất lượng cao sẽ cần đến một chi phí khổng lồ. Hiện tại thì có khá nhiều thuật toán nén ảnh ngày nay và thuật toán Jpeg là một trong số chúng.

**2. Đối tượng nghiên cứu**

Đồ án chủ yếu nghiên cứu về thuật toán JPEG (**Joint Photographic Experts Group) bao gồm cả thuật toán DCT (Discrete Cosine Transformation) thuật toán Huffman dung để nén dữ liệu bức ảnh mà mức ảnh hưởng tới chất lượng hình ảnh được nén không ảnh hưởng quá nhiều.**

**3. Cơ cấu đồ án**

       Nội dung của bản báo cáo gồm 3 phần:

       -Phần 1: Khái quát chung về đề tài

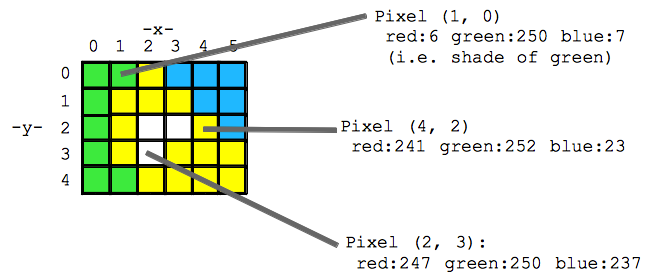
-Phần 2: Nén ảnh theo chuẩn JPEG

       -Phần 3: Trình bày Demo

Phần 2: Nén ảnh theo chuẩn JPEG

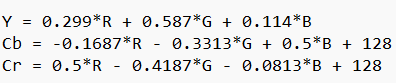
**I.**              **Chuẩn bị ảnh và Chrominance Conversion:**

## 1. Chuẩn bị ảnh:

Một bức ảnh gốc sẽ bao gồm nhiều pixel và mỗi pixel sẽ có thành phần Red(R), Blue(B), Green(G) với mỗi thành phần có giá trị từ 0 tới 255 và sự kết hợp của 3 thành phần đó sẽ tạo nên màu sắc cho từng pixel trong bức ảnh.

*1.a) hình ảnh minh họa cho một pixel*

Quá trình chuẩn bị ảnh lấy giá trị của 3 thành phần R,G,B để tính giá trị cho 3 thành phần mới Y(Luminance), CR( Red Chrominance) và Cb (Blue Chrominance) theo công thức sau:



Với Y là đại diện cho độ sán g của một pixel

Cb đo lường sự khác biệt giữa màu xanh và đỏ

Cr đo lường sự khác biệt màu xanh lục và đỏ

Quá trình này có thể được đảo ngược ( ví dụ như có thể tính giá trị của thành phần R,G,B từ các thành phần Y,Cr,Cb) và không có bất kì dữ liệu nào bị mất đi trong quá trình chuyển đổi.

**2.              Chrominance Conversion:**

Vì sự nhạy cảm của hệ thống thị giác người đối với chi tiết màu sắc so với chi tiết độ sang nên trong bước này ta sẽ lấy 2 thành phần Blue Chrominance và Red Chrominance và đối với mỗi khối 2x2 pixel, ta sẽ tính giá trị trung bình của giá trị Cb và Cr và lưu trữ cho toàn bộ khối đó ( có kích thước là 2x2), sau khi thực hiện bước này thì ta sẽ có những thành phần mà mắt người khó nhận biết ( là Cr và Cb) giảm còn ¼ kích thước ban đầu của nó, thành phần Y vẫn giữ nguyên( Chú ý khi tổng hợp các thành phần lại thì Cb và Cr sẽ được điều chỉnh kích thước bằng với thành phần Y trong bức ảnh sau đó ta có thể tính ngược lại các giá trị R,G,B theo công thức ở trong phần chuẩn bị ảnh ở trên và do thành phần Y thay đổi từ pixel sang pixel, những thành phần R,G,B mới được tính toán cũng sẽ thay đổi từ pixel sang pixel ).

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Cr:138 | |
|  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Cr:124 | Cr:140 |
|  |  | Cr:152 | Cr:136 |
|  |  |  |  |
|  |  |  |  |

|  |  |
| --- | --- |
|  | Cb:92 |
|
|  |  |
|

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Cb:80 | Cb:91 |
|  |  | Cb:94 | Cb:103 |
|  |  |  |  |
|  |  |  |  |

*Hình 2.1: Trước khi tính trung bình Hình 2.2: Sau khi tính trung bình*

**II.**          **Thuật toán DCT và** **Quantization**

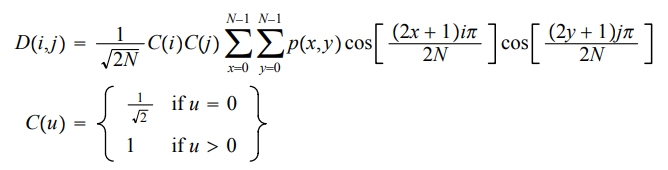
1. **Thuật toán DCT:**

Về cơ bản thì trong bước này nó sẽ quét toàn bộ bức ảnh và loại bỏ thành phần có tần số cao do những thành phần đó mắt khó có thể nhận biết.

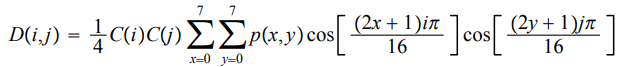
Cách DCT xử lý ở 3 thành phần Cb,Cr,Y đều là như nhau.

Bước đầu tiên của DCT là chia toàn bộ bức ảnh thành các ô có kích thước 8x8 pixels, với pixel trong đó sẽ có giá trị từ 0 đến 255 biểu thị cho độ chói sáng của từng pixels, sau đó ta sẽ thực hiện bước thuật toán DCT.

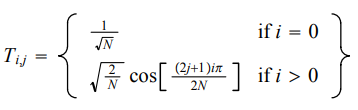
Phương trình DCT có dạng như sau:



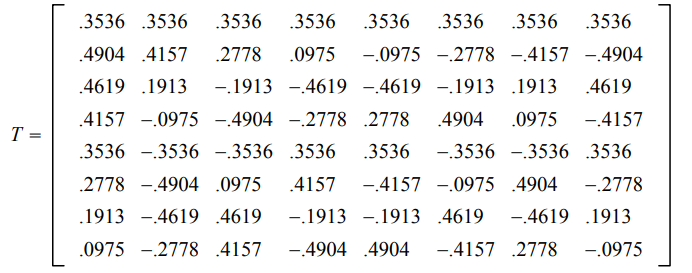
Với p(x,y) gồm x,y là vị trí của thành phần trong ma trận p. N là kích thước của ô được áp dụng DCT. Phương trình tính toán một entry (I,j)th của hình ảnh được chuyển đổi từ các giá trị pixels của ma trận hình ảnh gốc. trong trường hợp này ta dung ô có kích thước 8x8 nên N = 8 còn x và y sẽ chạy từ 0 tới 7 vì vậy ta sẽ có phương tình mới.



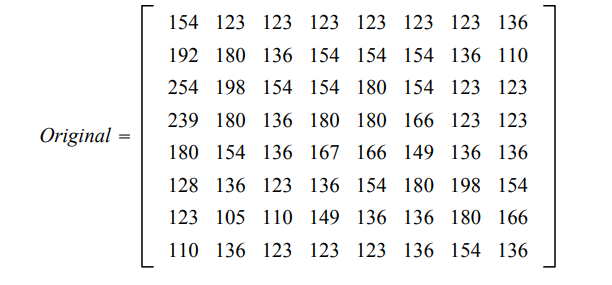
Để lấy dạng ma trận của phương trình D(I,j), ta dung phương tình sau;

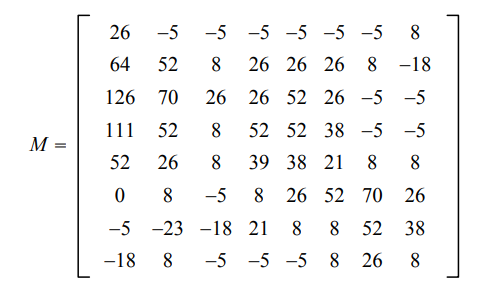


Áp dụng phương trình trên thì ta sẽ có một ma trận ví dụ cho ô 8x8:

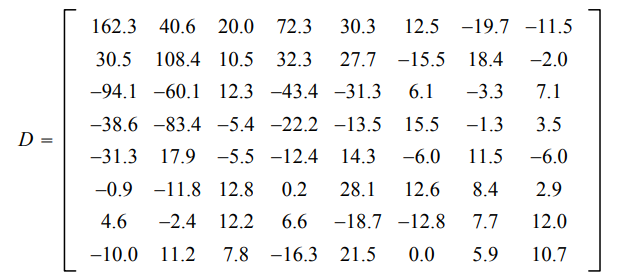


Sau đó thì ta trừ toàn bộ giá trị trong thành phần Y đi 128, từ đó thì ta có mỗi pixel trong thành phần Y sẽ có giá trị từ -128 đến 128 với -128 là màu đen còn 127 là màu trắng (Lý do thực hiện bước này là do thuật toán DCT được thiết kế hoạt động trên pixel có giá trị từ -128 đến 127).





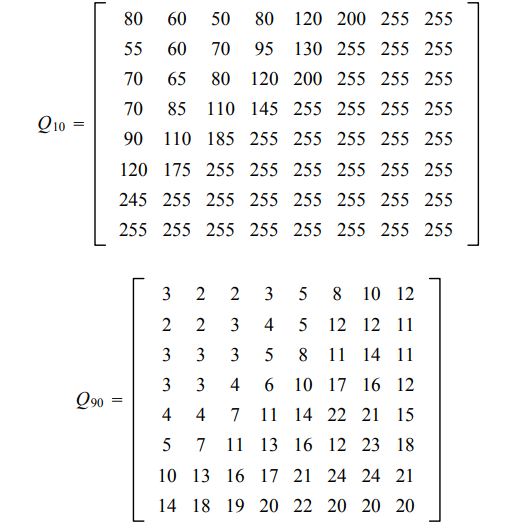
Ta cũng có thể thực hiện DCT bằng công thức sau: D = T\*M\*T’

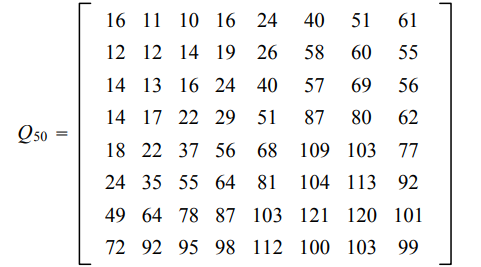


Ta có thể thấy phần tử ở vị trí [0,0] trong ma trận tương quan với tần số thấp của ô trong hình ảnh gốc. Khi mà ta di chuyển từ vị trí [0,0] từ mọi hướng, những hệ số DCT có xu hướng luôn cao hơn, với phần tử ở vị trí [7,7] có tần số cao nhất. Chú ý là mắt người là mắt người rất nhạy với ô có tần số thấp.

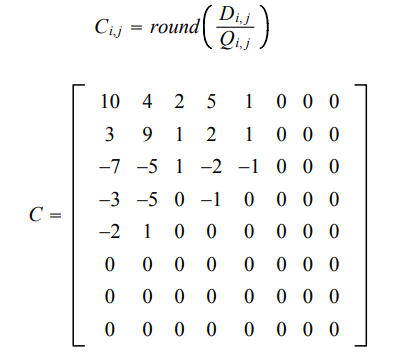
1. **Quantization:**

Trong bước này thì người dung có thể quyết định chất lương hình ảnh có các cấp độ từ 1 tới 100, với cấp 1 cho chất lượng hình ảnh thấp nhất và mức nén cao nhất, trong khi 100 cho chất lượng hình ảnh cao nhất và mức nén thấp nhất, đây là một ví dụ cho bảng quantization ở mức độ 50,10,90:





Việc quantization được thực hiện bằng cách chia mỗi phần tử của ma trân hình ảnh D cho từng phần tử tương ứng của ma trận quantization , và sau đó làm tròn lên số nguyên gần nhất của nó ta sẽ có kết quả như sau:



Ở đây ta thấy rằng giá trị gần góc trái trên cùng sẽ tương ứng với tần số thấp (mắt người đặc biệt nhạy cảm với loại ô hình ảnh này). Ngược lại thì những ô có hệ số 0 thì ít quan trọng hơn, hệ số tần số cao đã được loại bỏ và làm mất đi dung lượng cho bức ảnh gốc vì chỉ những hệ số khác 0 trong ma trận mới được dung để xây dựng lại hình ảnh sau khi nén. Đặc biệt chú ý là đối với ma trận Q20 thì có số lượng hệ số 0 nhiều hơn trong khi ma trận Q90 sẽ có ít số lượng hệ số 0.

**III.**          **Thuật toán RLE và Huffman**

1. **Thuật toán RLE (Run Length encoding)**

Thuật toán RLE là một dạng nén không mất mát. Thuật toán hoạt động bằng cách tìm loạt dữ liệu liền nhau, lặp lại trong chuỗi dữ liệu thành một dữ liệu đại diện khác, mục đích là để giảm kích thước dữ liệu gốc

# C:\Users\Admin\OneDrive\Máy tính\CS232.N21.KHCL\Final\Run-lengthEncoding1.png

*Hình 1.a: Hình ảnh minh họa cách hoạt động của RLE*

1. **Thuật toán Huffman coding**

Thuật toán Huffman coding là thuật toán nén không mất mát, Thuật toán có mục tiêu sẽ xây dựng được bảng mã nhị phân đại diện cho từng kí tự sao cho những kí tự có tần suất xuất hiện nhiều sẽ có mã nhị phân đại diện cho nó ngắn và ngược lại.

Thuật toán Huffman coding gồm 3 bước:

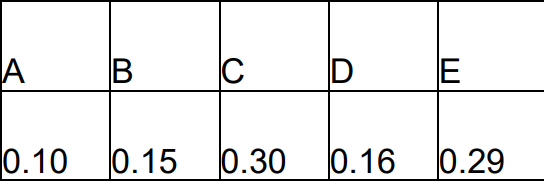
* Bước 1: Đếm tần suất xuất hiện của các phần tử trong chuỗi đầu vào.
* Bước 2: Xây dựng cây Huffman (cây nhị phân mã hóa).
* Bước 3: Từ cây Huffman, ta có được các giá trị mã hóa. Lúc này, ta có thể xây dựng chuỗi mã hóa từ các giá trị này.

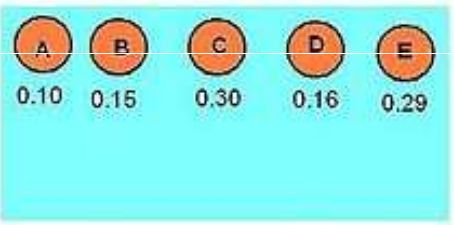
**Ý tưởng xây dựng mã Huffman:**

* Xây dựng bảng thống kê tần số xuất hiện của các ký tự cần mã hóa.
* Mỗi phần tử được xem như là đỉnh của một cây.
* Lặp lại cho tới lúc chỉ còn một cây (Chọn 2 cây có trọng số bé nhất ghép thành một cây mới)
* Từ đỉnh duyệt cây:
  + Nếu về bên trái chọn bit 0
  + Về phải chọn bit 1
  + Đến lá thì dãy bit đã duyệt chính là mã mới của ký tự

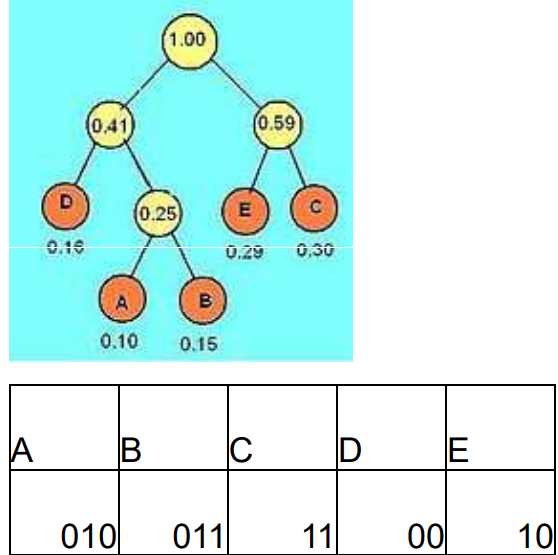
Ví dụ minh họa :

Ta có bảng tần số dưới



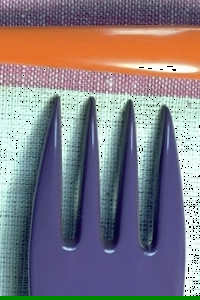
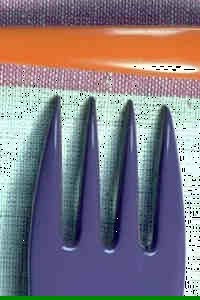
Cho mỗi phần tử là một đỉnh của cây, sau khi thực hiện tất cả các bước trên thì ta sẽ có cây mới và bảng mã mới.

*Hình 2.a: Mỗi phần tử là đỉnh của một cây.*



*Hình 2.b: Sau khi thực hiện thuật toán cho cây mới và bảng mã.*

# Phần 3: Kết quả và đánh giá



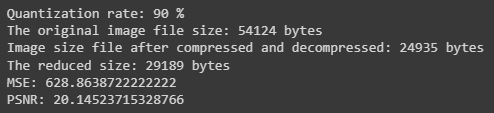
*Hình ảnh gốc Hình 3.1 Hình 3.2*

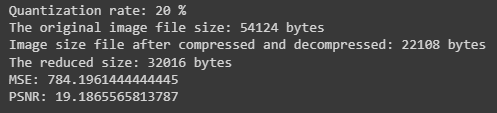
Với Hình 3.a là hình ảnh sau khi nén với Quantization rate = 90%

Hình 3.b là hình ảnh sau khi nén với Quantization rate = 20%

Sử dụng các phương pháp đánh giá MSE và PSNR để đánh giá mức độ nén của thuật toán:

Đánh giá hiệu quả sau khi nén với Quantization rate = 90%:



Đánh giá hiệu quả sau khi nén với Quantization rate = 20%:

**Tài liệu tham khảo**

1. JPEG compression explained: [https://www.baeldung.com/cs/jpeg-compression#:~:text=JPEG%20stands%20for%20Joint%20Photographic,smaller%20than%20the%20original%20one.](https://www.baeldung.com/cs/jpeg-compression%23:~:text=JPEG%20stands%20for%20Joint%20Photographic,smaller%20than%20the%20original%20one.)

2. Image data uncompressed: <https://www.reasoft.com/tutorials/web/plastic.html>

3. DTC and Quantization in image compression: <https://www.math.cuhk.edu.hk/~lmlui/dct.pdf>

4. How jpeg compression work: <https://medium.com/analytics-vidhya/what-are-dct-coefficients-and-how-jpeg-compression-works-7f46d1e22b4c>