| | |
|---|---|
| **CS 681: Computational Number Theory and Algebra** | **Lecture 7** |
| **Lecturer: Manindra Agrawal** | **Notes by: Zahir Koradia** |
| | **August 16, 2005.** |

# 1 Real life application of FFT: Image compression in JPEG format

JPEG performs DFT in order to compress the input file. In order to perform DFT we need an $n^{th}$ root of unity. The $n^{th}$ root of unity the we will use here is $e^{2\pi i/n}$. But the problem with this is that using this root the values obtained after performing DFT are complex, storing which is not space efficient. Hence, we use a trick to make sure all values of DFT are real meaning complex part of DFT is zero. Let $f : \{0, \ldots, n-1\} \longrightarrow F$. Assume $n$ is even.
Let $\tilde{f} : \{0. \ldots, 4n-1\} \longrightarrow F$ be defined as:
1. $\tilde{f}(2l) = 0 \; for \; 0 \leq l \leq 2n$
2. $\tilde{f}(2l+1) = f(l) = \tilde{f}(4n-(2l+1)) \; for \; 0 \leq l < n$

To make the definition more clear $\tilde{f}(l)$ is zero at all even values of l and for all the 2n add values of l we have: $\tilde{f}(1) = \tilde{f}(4n-1) = f(0), \tilde{f}(3) = \tilde{f}(4n-3) = f(1)$ and so on.
Now let $\tilde{g}(j) = \mathrm{DFT}(\tilde{f})$. Therefore,

$$\begin{aligned}
\tilde{g}(j) &= \textstyle\sum_{l=0}^{4n-1} \tilde{f}(l)\omega^{lj} \\
&= \textstyle\sum_{l=0}^{4n-1} \tilde{f}(l)e^{2\pi ilj/4n} \\
&= \textstyle\sum_{l_odd=0}^{4n-1} \tilde{f}(l)e^{2\pi ilj/4n} \\
&= \textstyle\sum_{l_odd=0}^{2n-1} \tilde{f}(l)e^{2\pi ilj/4n} + \sum_{l_odd=2n}^{4n-1} \tilde{f}(l)e^{2\pi ilj/4n} \\
&= \textstyle\sum_{l=0}^{n} \tilde{f}(2l+1)e^{2\pi ij(2l+1)/4n} + \sum_{l=0}^{n} \tilde{f}(4n-2l-1)e^{2\pi ij(4n-2l-1)/4n} \\
&= \textstyle\sum_{l=0}^{n} \tilde{f}(l)[e^{2\pi ij(2l+1)/4n} + e^{2\pi ij(4n-2l-1)/4n}] \\
&= \textstyle\sum_{l=0}^{n} \tilde{f}(l)2\cos(\frac{2\pi j(2l+1)}{4n})
\end{aligned}$$

The mapping $f \longrightarrow \tilde{g}$ is called discrete cosine transform. Note that:
1. $\tilde{g}(4n-j) = \tilde{g}(j)$ for $0 \leq j < n$
2. $\tilde{g}(2n+j) = \tilde{g}(2n-j) = \tilde{g}(j)$ for $0 \leq j < n$

This shows that it is enough to store the first $n$ values of the discrete cosine transform. The inverse of $\tilde{g}$ can then be computed as:

$$\tilde{f}(j) = \sum_{l=0}^{4n-1} \tilde{g}(l)e^{2\pi ijl/4n}$$

## 1.1 Compression using discrete cosine transform:

Let $I0, \ldots, n-1\text{x}0, \ldots, n-1 \longrightarrow N$ be an image.

### 1.1.1 First Algorithm:

1. For each row of $I$ do the following.
2. Apply DCT on the row to obtain n values $c_0, c_1, \ldots, c_{n-1}$. These values can be understood to be coefficients of frequencies contributing to the image. If $c_j$ is very small then associated frequency contributes very little and we can simply make $c_j$ equal to 0.
3. Let $d_j = \lfloor \frac{c_j}{100} \rfloor$
4. Perform runlength encoding or huffman encoding or both on $d_0, d_1, \ldots, d_{n-1}$ and store the result.

### 1.1.2 Secong Algorithm:

1. Break the image into bloack of (say) 8x8 pexels.
2. Take each two-dimensional block and apply DCT using $g(l,t) = \sum_j \sum_k f(j,k)\omega^{lj}\omega^{tk}$

JPEG normally gives a 20:1 ratio compression.