Experiment 4: Image Transformation using MATLAB

By: Dr. Rajeev Srivastava

1. **Discrete Fourier Transform (DFT)**

The 2-D discrete Fourier transform of function $f$ denoted by $F(u, v)$ is given by:

$$ F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi \frac{ux}{M} \frac{vy}{N}} $$

for $u=0, 1, 2... M-1$ and $v=0, 1, 2... N-1$

The values $F(u,v)$ are the DFT coefficients of $f(x,y)$.

The zero-frequency coefficients, $F(0,0)$ is often called the ‘DC component’.

DFT of a vector $x$ of length $n$ is another vector $y$ of length $n$ is given by:

$$ y_{p+1} = \sum_{j=0}^{n-1} \omega^{jp} x_{j+1} $$

where $\omega$ is a complex $n^{th}$ root of unity,

$$ \omega = e^{-\frac{2\pi}{n}} $$

The MATLAB functions `fft`, `fft2` and `fftn` and their inverses `ifft`, `ifft2` and `ifftn` all use Fast Fourier Transform algorithm to compute DFT.

![Fig: FFT Transformation](image)

2. **Discrete Cosine Transform (DCT)**

DCT represents an image as a sum of sinusoids of varying magnitude and frequencies. DCT has the property that most of the information about the image is concentrated in just a few coefficients of DCT. For this reason DCT is used in compression.

**Example:** DCT is at the heart of the international standard lossy image compression algorithm known as JPEG (Joint Photographic Expert Group).

- $Y = dct(X)$ returns the discrete cosine transform of $X$.

$$ Y(k) = w(k) \sum_{n=1}^{N} x(n) \left( \frac{\cos(2\pi n - 1)(k-1)}{2N} \right) $$

Where $K=1, 2, ..., N$
\[ w(k) = \frac{1}{\sqrt{N}}, k = 1 \]
\[ = \frac{2}{\sqrt{N}}, 2 \leq k \leq N \]

N is the same length of X and X & Y are of same size.
\[ Y=dct(X,n) \text{ pads or truncates } x \text{ to length } n \text{ before transforming.} \]

- There are two ways to compute DCT using image processing toolbox. 1st method is to use \texttt{dct2} function. \texttt{dct2} uses an FFT-based algorithm for speedy computation with large inputs. The second method is to use the DCT transform matrix, which is returned by the function \texttt{dctmtx} and might be more efficient for small square inputs, such as 8-by-8 or 16-by-16.

3. WALSH HADAMARD TRANSFORM

Walsh Hadamard transform is a non-sinusoidal orthogonal transform technique that decomposes a signal into set of basic functions. These functions are Walsh functions which are rectangular or square waves with +1 or -1.

- **Hadamard function**:
  \[ H=hadamard(n) \text{ returns hadamard matrix of order } n. (n \text{ must be an integer and } n,n/12 \text{ or } n/20 \text{ must be a power of 2).} \]
  Hadamard matrices are matrices of 1’s and -1’s whose columns are orthogonal.
  \[ H' \cdot H = n \cdot I \]
  Where \( [n n]=size(H) \text{ and } I=eye(n,n); \text{ (eye returns the identity matrix)} \)

- Different ordering scheme are used to store Walsh function. That are:
  - Sequency: coefficients in order of increasing sequence value.
  - Hadamard: coefficient in hadamard order.
  - Dyadic: coefficient in gray code order.

- Application of Walsh hadamard transform: speech processing, filtering and power spectrum analysis.
- Walsh-hadamard transform has a fast version \texttt{fwht} (fast Walsh-hadamard transform). Compare to \texttt{fft}, \texttt{fwht} requires less storage space and is faster to calculate because it uses real addition and subtraction while \texttt{fft} requires complex calculation. Inverse of \texttt{fwht} is \texttt{ifwht}.

- **Definition of FWHT**:
  \[ Y_n = \frac{1}{N} \sum_{i=0}^{N-1} x, WAL(n, i) \]

- **Definition of IFWHT**: \[ x_j = \sum_{i=0}^{N-1} Y_n \cdot WAL(n, i) \]
- \[ Y=fwht(X) \text{ returns the coefficient of the discrete Walsh Hadamard transform.} \]
- \[ Y=fwht(X,n) \text{ returns the } n \text{-point discrete Walsh hadamard transform, } n\text{must be power of 2. } X \text{ and } n \text{ must be of same length. If } X \text{ is longer than } n, X \text{ is truncate and if } X \text{ is shorter than } n, X \text{ is padded with zeros.} \]
- \[ Y=fwht(X,n,ordering) \text{ specifies the ordering to use for the returned Walsh-hadamard transform coefficient.} \]
4. **Radon transform**

The Radon transform of a two dimensional function, $f(x,y)$ is the line integral of $f$ parallel to the $y'$-axis given by:

$$R_\theta(x') = \int_{-\infty}^{\infty} f(x' \cos \theta - y' \sin \theta, x' \sin \theta + y' \cos \theta) \, dy'$$

where

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

The radon function computes projections of an image matrix along specified directions. It computes the line integral from multiple source along parallel paths, or beams, in a certain direction. The beams are spaced 1 pixel unit apart. To represent an image, radon function takes multiple, parallel-beam projections of the image from different angles by rotating the source around the center of the image.

Its syntax is:

```matlab
R=radon(I, theta)
```

This function returns the Radon transform $r$ of the intensity image $I$ for the angle $\theta$ degrees.