Discrete Cosine Transform and JPEG

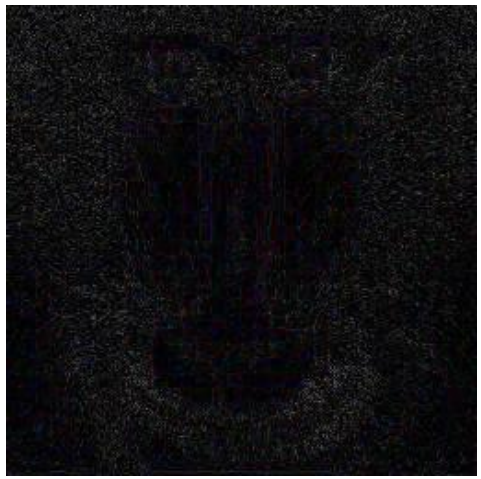
L. Karlsson

Example

(a) Original

(b) Low

(c) High



(d) Error (high)

Part I

Images

Colors

- RGB:

$$\mathbf{p}_{\mathrm{RGB}} = \begin{pmatrix} r \\ g \\ b \end{pmatrix},$$

  where r, g, b are the red, green, and blue components of the pixel $\mathbf{p}$.

- YCbCr:

$$\mathbf{p}_{\mathrm{YCbCr}} = \begin{pmatrix} y \\ c_{\mathrm{b}} \\ c_{\mathrm{r}} \end{pmatrix},$$

  where y is the luminance, and c_{b} and c_{r} are the chroma components.

- RGB and YCbCr are related by

$$\mathbf{p}_{\mathrm{YCbCr}} = \begin{pmatrix} 0.2990 & 0.5870 & 0.1140 \\ -0.1687 & -0.3313 & 0.5000 \\ 0.5000 & -0.4187 & -0.0813 \end{pmatrix} \mathbf{p}_{\mathrm{RGB}} + \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix}.$$

Images

MATLAB commands:

- Load an image: `A = imread(filename)`.
  - `A` is an $m \times n \times 3$ array of type `uint8` (integers 0–255).
  - The pixels of `A` are in the RGB format.
  - `A(i,j,:)` is the pixel on row i and column j.
  - `A(:,:,2)` is the green component of each pixel.
- Display an image: `image(A); axis equal; axis off`.

Numerical types

- The function `image` expects a matrix of the `uint8` type.
- Floating point numbers in MATLAB are of type `double`.
- Convert to specific type using the type name as a function.
  - To `double`: `B = double(A)`.
  - To `uint8`: `A = uint8(B)`. Values outside 0–255 are forced to the nearest bound.
- Expressions with mixed types often cause errors.
- Note that `uint8(2) - uint8(5)` is `0` of type `uint8`. Convert to `double` before subtracting if you want the result `-3`.

Conversion between RGB and YCbCr

MATLAB commands:

- RGB to YCbCr: `B = rgb2ycbcr(A)`
- YCbCr to RGB: `A = ycbcr2rgb(A)`

Part II

Discrete Cosine Transform (DCT)

1-D DCT

Component-wise form

- Input: vector $\mathbf{x} \in \mathbb{R}^n$.
- Output: vector $\mathbf{y} \in \mathbb{R}^n$.
- Transform in component-wise form:

$$y_u = C(u)\sqrt{\frac{2}{n}}\sum_{i=1}^{n} x_i \cos\left(\frac{(2i-1)(u-1)\pi}{2n}\right),$$

where

$$C(u)\begin{cases} 1/\sqrt{2} & \text{if } u = 1, \\ 1 & \text{otherwise.} \end{cases}$$

1-D DCT

Matrix form

- Input: vector $\mathbf{x} \in \mathbb{R}^n$.
- Output: vector $\mathbf{y} \in \mathbb{R}^n$.
- Transform in matrix form:

$$\mathbf{y} = Z\mathbf{x},$$

where

$$Z \in \mathbb{R}^{n \times n} \text{ and } z_{ij} = C(i)\sqrt{\frac{2}{n}} \cos\left(\frac{(2j-1)(i-1)\pi}{2n}\right).$$

- In MATLAB: `Z = dctmtx(n)`.
- Z is orthogonal, i.e., $Z^{-1} = Z^T$.

2-D DCT

(Forward) DCT:

- Input: matrix $X \in \mathbb{R}^{n \times n}$.
- Output: matrix $Y \in \mathbb{R}^{n \times n}$.
- Transform in matrix form:

$$Y = ZXZ^T.$$

- In MATLAB: `Y = dct2(X)`.

Inverse DCT:

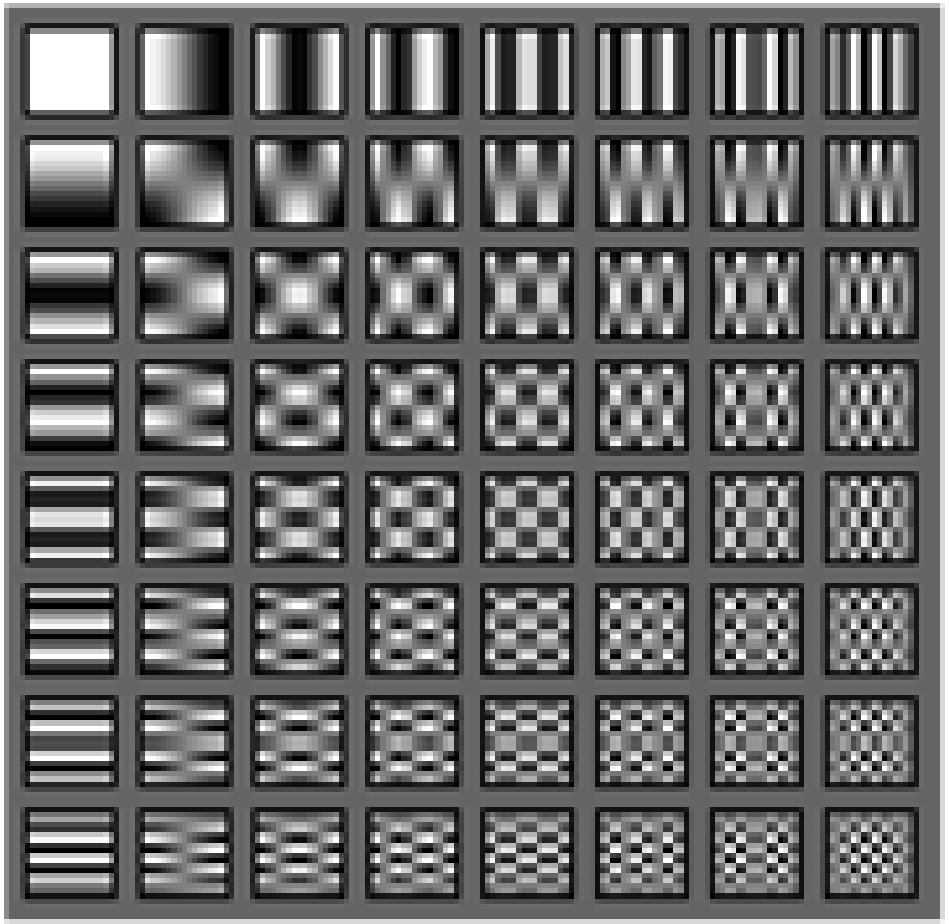
- Input: matrix $Y \in \mathbb{R}^{n \times n}$.
- Output: matrix $X \in \mathbb{R}^{n \times n}$.
- Transform in matrix form:

$$X = Z^T YZ.$$

- In MATLAB: `X = idct2(Y)`.

DCT basis functions

$$B_{ij} = Z(e_i e_j^T) Z^T$$



Part III

JPEG

JPEG

- *Joint Photographic Experts Group* (JPEG)
- A continuous tone image compression standard.
- Builds on the 2-D DCT of size 8×8.
- Uses YCbCr images, treated as three separate images.

JPEG compression algorithm

- Convert the image to YCbCr.
- Partition the image into blocks of size 8×8.
- For each component of each block:
  - Apply the 2-D DCT.
  - Quantize (discards information).
- Encode (compresses).

JPEG decompression algorithm

- Decode (decompresses).
- Partition the image into blocks of size 8×8.
- For each component of each block:
  - Dequantize.
  - Apply the 2-D inverse DCT.
- Convert the image to RGB.

Quantization

- Input: matrix $X \in \mathbb{R}^{n \times n}$ and quantization table $Q \in \mathbb{Z}^{n \times n}$.
- Output: quantized matrix $Y \in \mathbb{Z}^{n \times n}$.
- Transformation in component-wise form:

$$y_{ij} = \text{round}(x_{ij}/q_{ij}).$$

Dequantization

- Input: quantized matrix $Y \in \mathbb{Z}^{n \times n}$ and quantization table $Q \in \mathbb{Z}^{n \times n}$.
- Output: dequantized matrix $X \in \mathbb{Z}^{n \times n}$.
- Transformation in component-wise form:

$$x_{ij} = y_{ij} q_{ij}.$$