Multimedia Engineering

Programming #2: DCT

Definition of DCT

- Given an input function $f(i, j)$ over two integer variables i and j (a piece of an image), the 2D DCT transforms it into a new function $F(u, v)$, with integer u and v running over the same range as i and j. The general definition of the transform is

$$F(u,v) = \frac{2\,C(u)\,C(v)}{\sqrt{MN}} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \cos\frac{(2i+1)\cdot u\pi}{2M} \cdot \cos\frac{(2j+1)\cdot v\pi}{2N} \cdot f(i,j)$$

- where i, u = 0, 1, … , M-1 (row); j, v = 0, 1, … , N-1 (col); and the constants $C(u)$ and $C(v)$ are determined by

$$where: C(u), C(v) = 1/\sqrt{2} \quad \text{for u, v} = 0$$

$$C(u), C(v) = 1 \quad \text{otherwise}$$

2D DCT & IDCT (DCT^{-1})

$$F(u,v) = \frac{C(u)\,C(v)}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} \cos\frac{(2i+1)u\pi}{16} \cos\frac{(2j+1)v\pi}{16} f(i,j)$$

$$\tilde{f}(i,j) = \sum_{u=0}^{7} \sum_{v=0}^{7} \frac{C(u)\,C(v)}{4} \cos\frac{(2i+1)u\pi}{16} \cos\frac{(2j+1)v\pi}{16} F(u,v)$$

- where $i, j, u, v = 0, 1, \ldots, 7$

1D DCT & IDCT (DCT^{-1})

$$F(u) = \frac{C(u)}{2} \sum_{i=0}^{7} \cos\frac{(2i+1)u\pi}{16} f(i)$$

$$\tilde{f}(i) = \sum_{u=0}^{7} \frac{C(u)}{2} \cos\frac{(2i+1)u\pi}{16} F(u)$$

- where $i, u = 0, 1, \ldots, 7$

2D Separable Basis

- The 2D DCT can be separated into a sequence of two 1D DCT steps

$$G(i,v) = \frac{1}{2}C(v)\sum_{j=0}^{7}\cos\frac{(2j+1)v\pi}{16}f(i,j)$$

$$F(u,v) = \frac{1}{2}C(u)\sum_{i=0}^{7}\cos\frac{(2i+1)u\pi}{16}G(i,v)$$

- It is straightforward to see that this simple change saves many arithmetic steps. The number of iterations required is reduced from 8x8 to 8+8

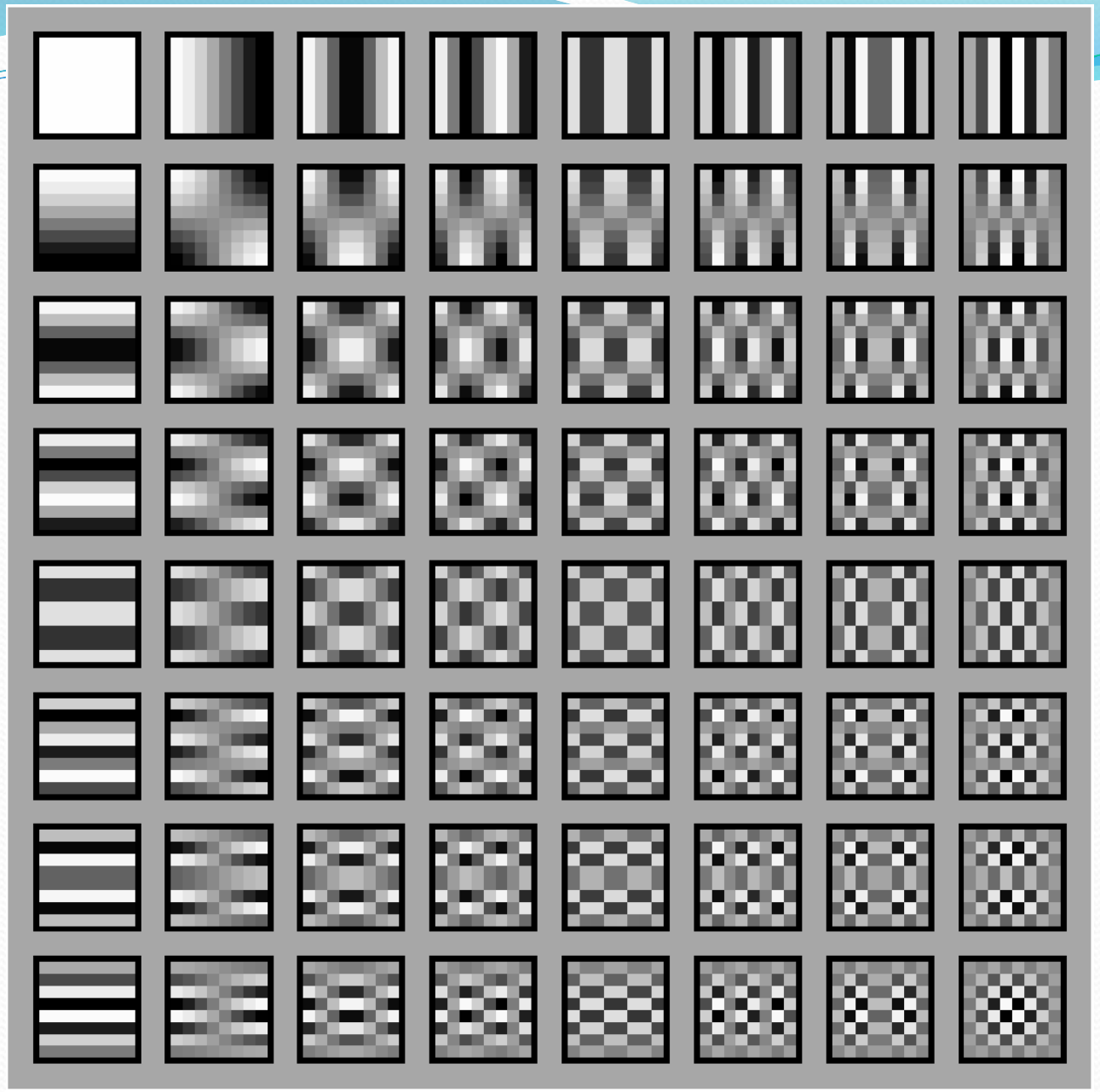

Fig. 8.9: Graphical Illustration of 8×8 2D DCT basis

Perform 2D DCT on 256x256 image

- 1st approach (given c-code)
  - 2D DCT on the whole image
  - Scale the resulting image to 0~255 appropriately
  - Display the resulting image *with log scale*
- 2nd approach
  - Step 1: 1D DCT on the lines of the image
  - Step 2: 1D DCT on the columns of the resulting image of Step 1
  - Scale the resulting image to 0~255 appropriately
  - Display the resulting image *with log scale*

- Comparison
  - Compare the resulting transformed image
  - Compare the speed of the transform
  - Perform 2D DCT over 3 different images(256x256) & Discuss the energy concentration of DCT transform

Sample Code for 256x256 DCT

```
#include <stdio.h>
#include <math.h>
#include <iostream>

#define M_PI 3.141592
#define C(u)  ((u)==0 ? (1/sqrt(2)) : 1)
#define min(x, y)              ((x)>(y) ? (y) : (x))
#define abs(x)                 ((x)>0 ? (x) : (-x))
#define max(x, y)              ((x)>(y) ? (x) : (y))

int main()
{
      unsigned char in_img[256][256];
      unsigned char out_img[256][256];
      char in_name[256], out_name[256];
      //int iblk[8][8], oblk[8][8];
      double oblk[256][256], tmp, max_tmp;
      int i, j;
      int          u, v;
      FILE* fpi, * fpo;

      printf("Input an image file name to process: ");
      scanf_s("%s", in_name, 256);

      printf("Input an output image file name to save: ");
      scanf_s("%s", out_name, 256);

      // read image file(img_name) in bit mode
      fopen_s(&fpi,in_name, "rb");
      if (fpi == NULL) {
                  printf("File %s open failure!!\n", in_name);
                  exit(0);
      }
```

```
	// write image file(img_name) in bit mode
	fopen_s(&fpo, out_name, "wb");
	if (fpo == NULL) {
		printf("File %s open failure!!\n", out_name);
		exit(0);
	}

	fread(in_img, 256 * 256, sizeof(unsigned char), fpi);

	for (u = 0; u < 256; u++)
		for (v = 0; v < 256; v++) {
		tmp = 0;
		for (i = 0; i < 256; i++)
			 for (j = 0; j < 256; j++)
			tmp += cos((2 * i + 1) * u * M_PI / (2 * 256)) * cos((2 * j + 1) * v * M_PI / (2 * 256)) *
	in_img[i][j];

		if (u == 0 & v == 0)
			oblk[u][v] = tmp / 256;
		else if (u == 0 || v == 0)
			oblk[u][v] = sqrt(2) * tmp / 256;
		else
			oblk[u][v] = 2 * tmp / 256;
	}

	max_tmp = 0;
	for (u = 0; u < 256; u++)
		for (v = 0; v < 256; v++)
			max_tmp = max(max_tmp, abs(oblk[u][v]));

	for (u = 0; u < 256; u++)
		for (v = 0; v < 256; v++)
			out_img[u][v] = min((int)(abs(oblk[u][v]) / max_tmp * 4096 * 2), 255);

	fwrite(out_img, 256 * 256, sizeof(unsigned char), fpo);

	fclose(fpi);
	fclose(fpo);
}
```

Result of 2D DCT

ORIGINAL

TRANSFORMED

Programming Homework #2

- Problem in page 6
- Due
  - 11/9 (Tue) 1:30 pm