**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ТВ**

**отчЁт**

**по лабораторной работе № 2**

**по дисциплине «Цифровая обработка изображений»**

**Тема: РАЗРАБОТКА ПРОГРАММЫ ДЛЯ**

**СПЕКТРАЛЬНОГО ПРЕДСТАВЛЕНИЯ СИГНАЛОВ**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 9105 |  | Шаривзянов Д. Р. |
|  |  | Басманов А. А. |
| Преподаватель |  | Поздеев А. А. |

Санкт-Петербург

2024

СПЕКТРАЛЬНОЕ ПРЕДСТАВЛЕНИЕ СИГНАЛОВ

**Цели работы:** знакомство со спектральным представлением сигналов и дискретным косинусным преобразованием (ДКП).

1. Код программы

struct Image{

Mat bgr;

Mat gray;

Mat full\_blocked;

// Mat selected\_block;

Mat image\_block;

int block\_size = 8;

};

Mat getBlocked(const Mat &img\_bgr, int block\_size){

Mat image\_with\_blocks = img\_bgr.clone();

for (int row = 0; row < img\_bgr.rows; row += block\_size){

for (int col = 0; col < img\_bgr.cols; col += block\_size){

cv::Rect block = cv::Rect(col, row, block\_size, block\_size);

cv::rectangle(image\_with\_blocks, block, cv::Scalar(0, 255, 0));

}

}

return image\_with\_blocks;

}

Mat getDCT(const Mat &image\_block){

int block\_size = image\_block.rows;

Mat basisMat = Mat::zeros(block\_size, block\_size, CV\_64F);

for (int row = 0; row < basisMat.rows; row++){

for (int col = 0; col < basisMat.cols; col++){

if (row == 0) basisMat.at<double>(row, col) = 1 / sqrt(block\_size);

else if (row > 0) basisMat.at<double>(row, col) = sqrt(2. / block\_size) \* cos(((CV\_PI\*row) / block\_size)\*(col + 0.5));

}

}

Mat imageBlock64F;

image\_block.convertTo(imageBlock64F, CV\_64F);

Mat DCT = basisMat \* imageBlock64F \* basisMat.t();

// std::cout << DCT << std::endl;

Mat DCT8U;

DCT.convertTo(DCT8U, CV\_8U);

Mat img\_dct = Mat::zeros(DCT8U.rows, DCT8U.cols, CV\_8UC1);

resize(DCT8U, img\_dct, cv::Size(400, 400), 0, 0, INTER\_NEAREST);

return img\_dct;

}

static void onMouse(int event, int x, int y, int, void\* param){

if (event != EVENT\_LBUTTONDOWN) return;

int blockX = x, blockY = y;

Image\* img = (Image\*)param; //получаем указатель на структуру изображения

for (int row = 0; row < img->full\_blocked.rows; row = row + img->block\_size){

for (int col = 0; col < img->full\_blocked.cols; col = col + img->block\_size){

if ((y >= row) && (y < row + img->block\_size) && (x >= col) && (x < col + img->block\_size)){

blockY = row;

blockX = col;

}

}

}

cout << blockX << " " << blockY << endl;

Rect block = Rect(blockX, blockY, img->block\_size, img->block\_size);

img->image\_block = img->gray(block);

Mat image\_block\_resized;

resize(img->image\_block, image\_block\_resized, cv::Size(400, 400), 0, 0, INTER\_NEAREST);

imshow("image block", image\_block\_resized);

imwrite("../../Images/Lab 2/image block.jpg", image\_block\_resized);

rectangle(img->full\_blocked, block, Scalar(0, 0, 255), 1);

imshow("image blocked", img->full\_blocked);

img->full\_blocked = getBlocked(img->full\_blocked, img->block\_size); //отчистим последний блок

Mat img\_dct = getDCT(img->image\_block);

imshow("image DCT", img\_dct);

imwrite("../../Images/Lab 2/image DCT.jpg", img\_dct);

return;

}

void lab2(const Mat &img\_bgr){

Image img;

img.bgr = img\_bgr;

imshow("image bgr", img.bgr);

cvtColor(img\_bgr, img.gray, COLOR\_BGR2GRAY);

imshow("image gray", img.gray);

imwrite("../../Images/Lab 2/image gray.jpg", img.gray);

// img.block\_size = 16;

img.full\_blocked = getBlocked(img\_bgr, img.block\_size);

imshow("image blocked", img.full\_blocked);

imwrite("../../Images/Lab 2/image blocked.jpg", img.full\_blocked);

setMouseCallback("image blocked", onMouse, (void\*)&img);

waitKey();

}

1. Исходное изображение и его гистограмма



Рис 1. Исходное полутоновое изображение.

1. Результат разбиения изображения на блоки



Рис. 2. Изображение, разбитое на блоки 8х8 пикс.

1. ДКП для разных «сюжетов»

Представлены результаты выбранного блока изображения (слева), увеличенный в масштабе блок (справа сверху) и результат ДКП (справа снизу).

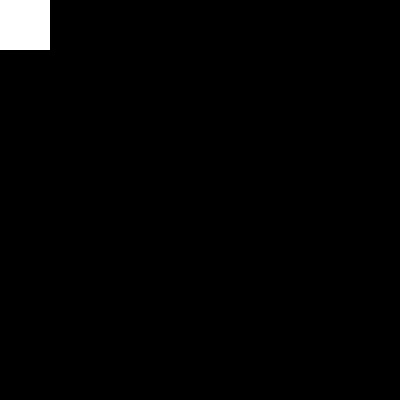


Рис. 3. Монотонный сюжет.

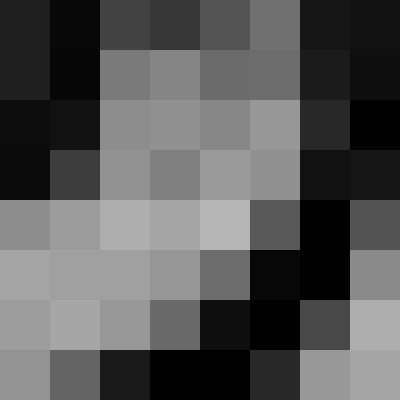
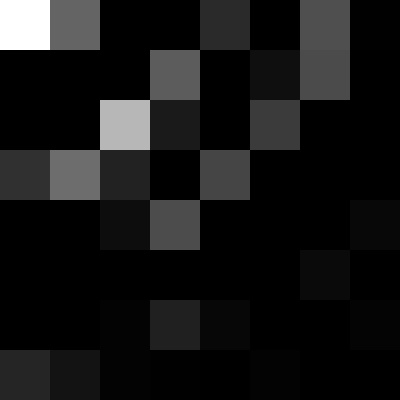


Рис. 4. Сюжет с резкими диагональными переходами.

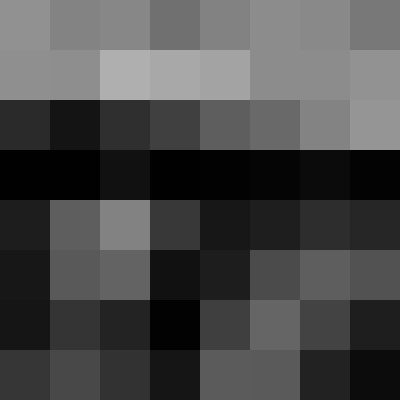


Рис. 5. Сюжет с резким вертикальным переходом.

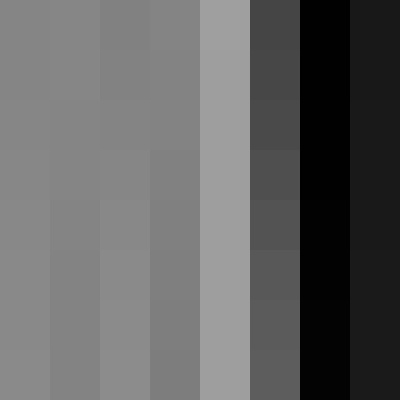


Рис. 6. Сюжет с резким горизонтальным переходом.

**Выводы:** Дискретное косинусное преобразование (ДКП) — это метод преобразования дискретных данных в комбинации косинусных волн. Применяя ДКП отдельно к каждому компоненту пикселя, можно получить получаем матрицу коэффициентов 8х8, которая показывает вклад каждой из 64 косинусной функции во входной матрице 8х8. Левый верхний угол представляет собой самую низкочастотную косинусную функцию, а правый нижний — самую высокочастотную. При монотонном сюжете в матрице коэффициентов ДКП самые большие значения (яркие на изображении) находятся в левом верхнем углу, а в случае сюжета с граничными переходами большие значения матрицы смежаются ближе к правому нижнему в зависимости от наклона границ и частоты переходов этих границ.

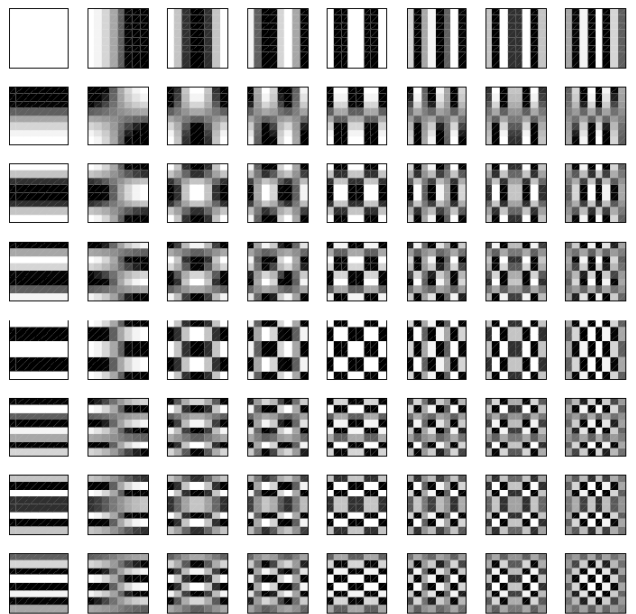


Рис. 7. Матрица базисных функций ДКП.