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();
}
```

2. Исходное изображение и его гистограмма

Рис 1. Исходное полутоновое изображение.

3. Результат разбиения изображения на блоки

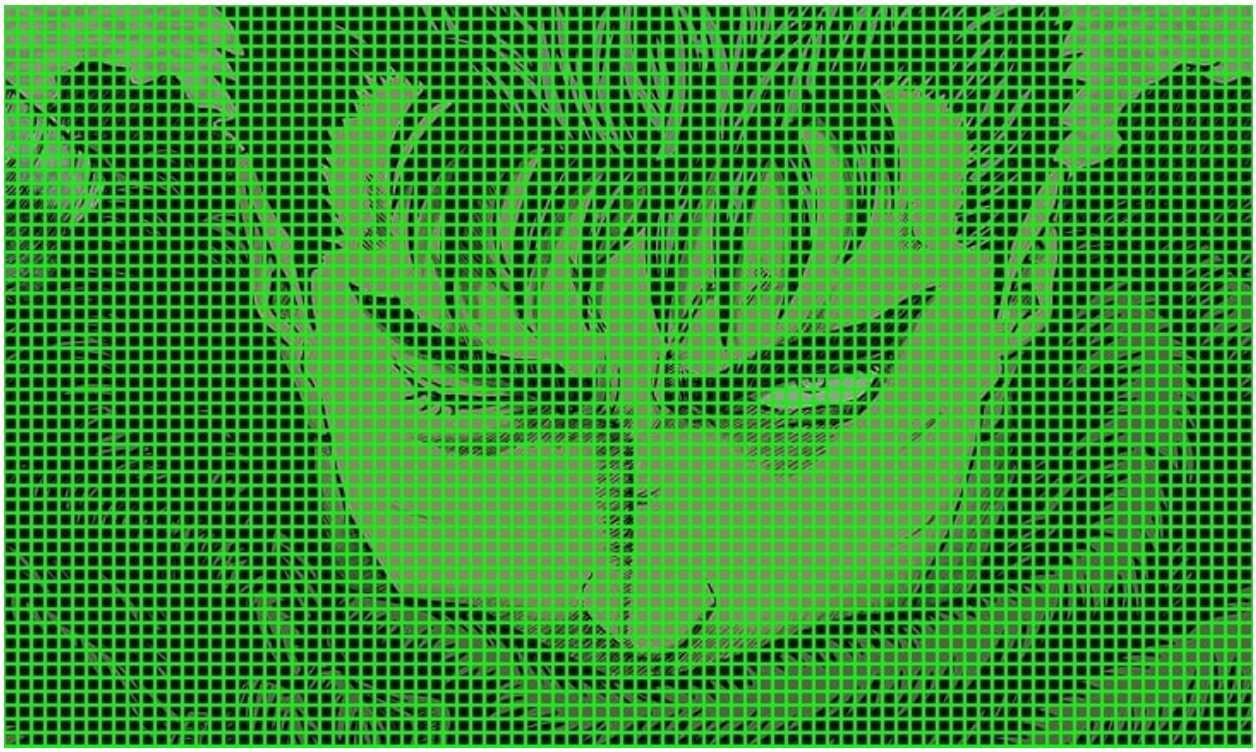

Рис. 2. Изображение, разбитое на блоки 8x8 пикс.

4. ДКП для разных «сюжетов»

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

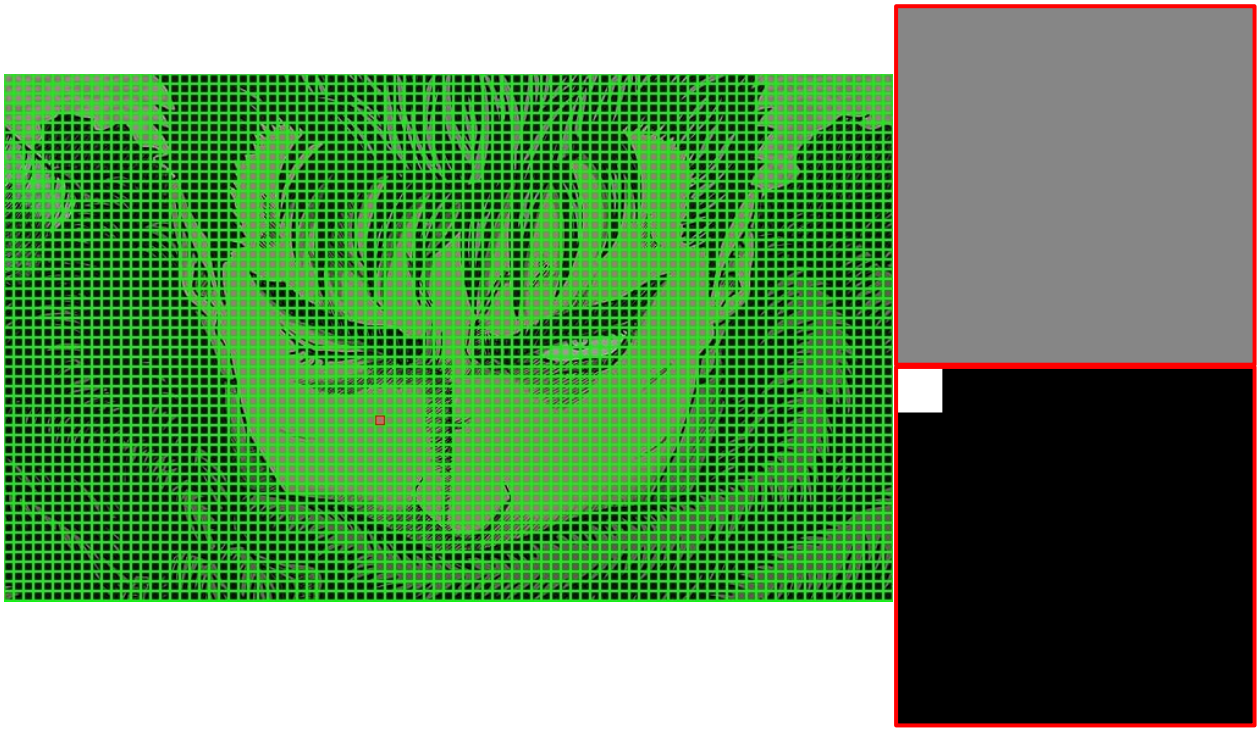

Рис. 3. Монотонный сюжет.

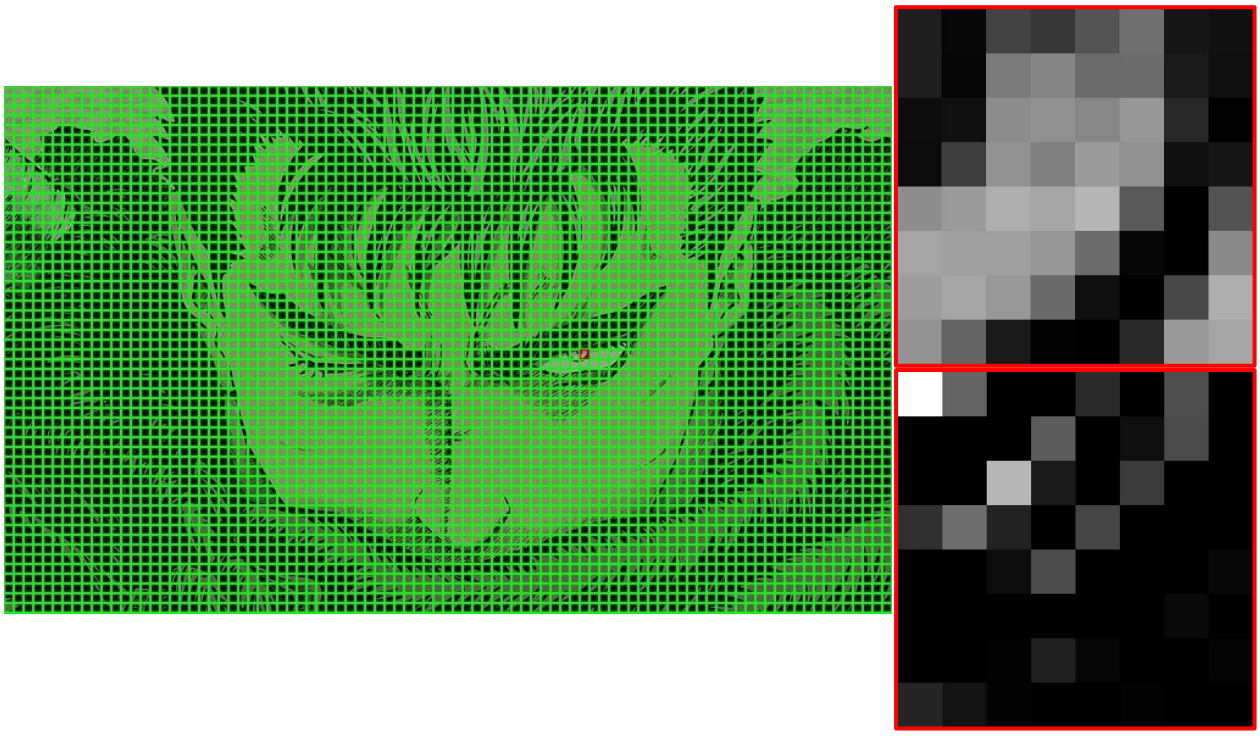

Рис. 4. Сюжет с резкими диагональными переходами.

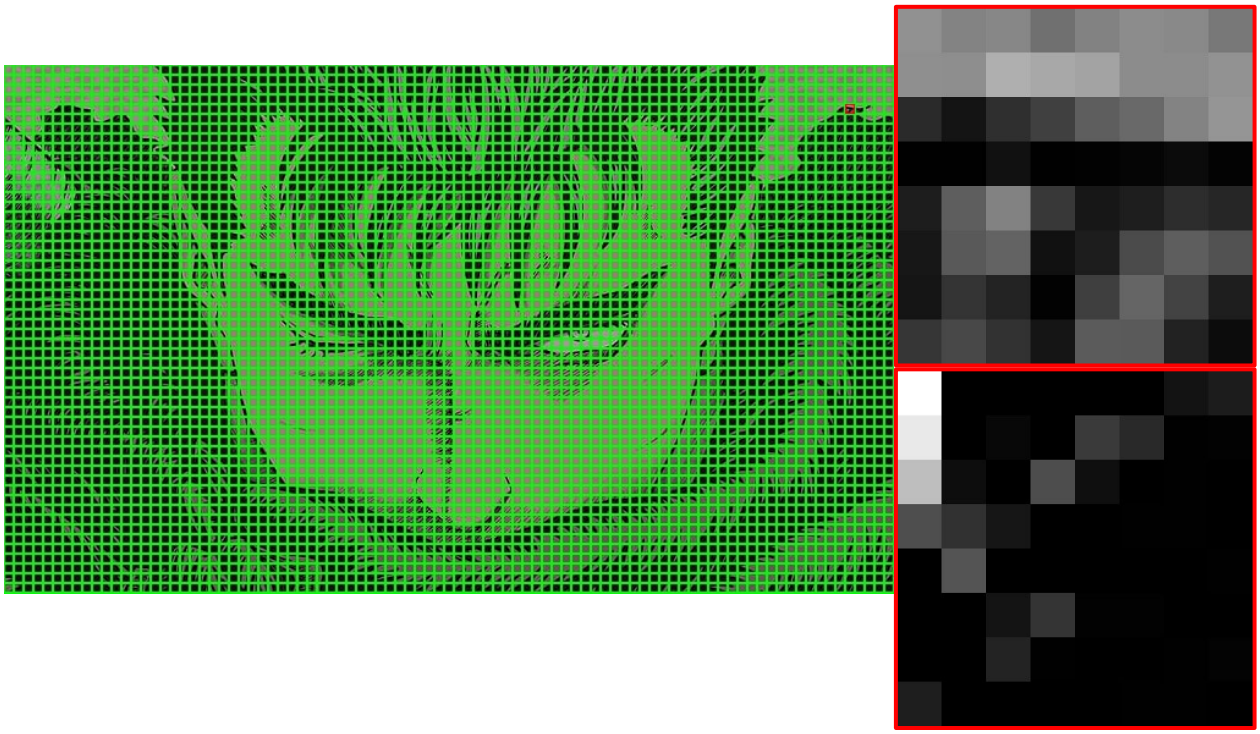

Рис. 5. Сюжет с резким вертикальным переходом.

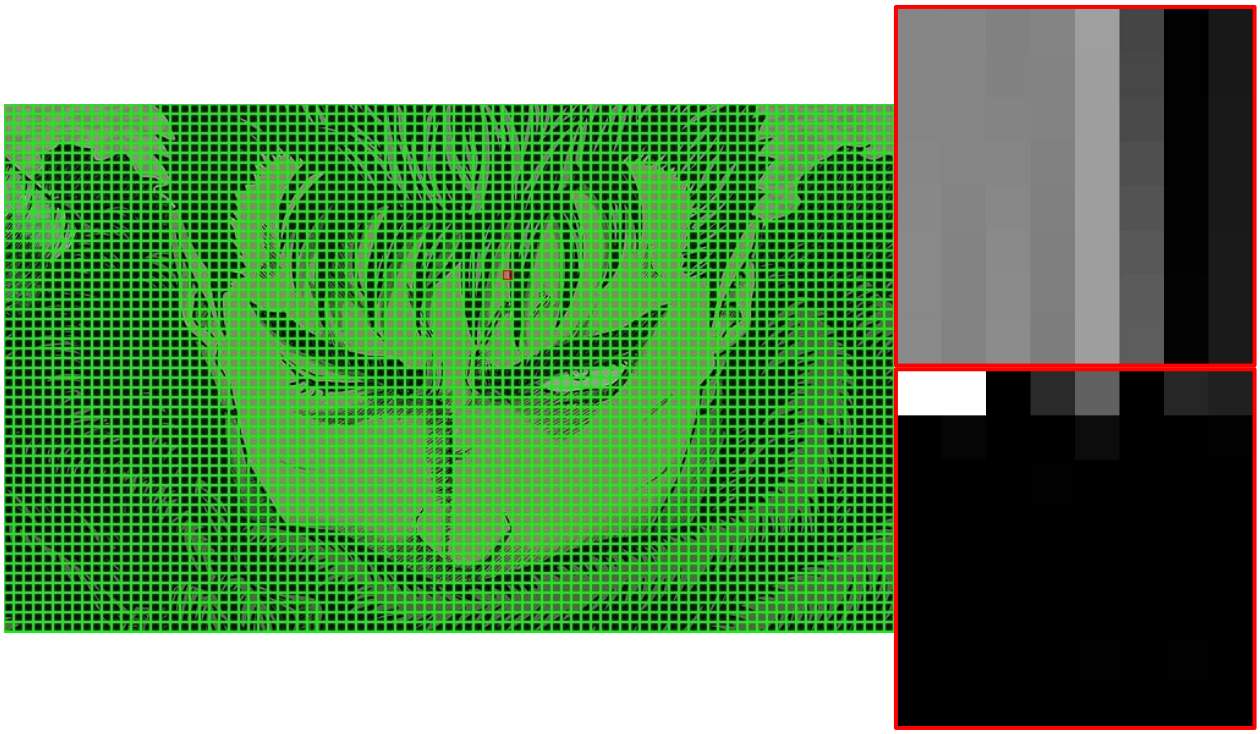

Рис. 6. Сюжет с резким горизонтальным переходом.

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 8x8, которая показывает вклад каждой из 64 косинусной функции во входной матрице 8x8. 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.

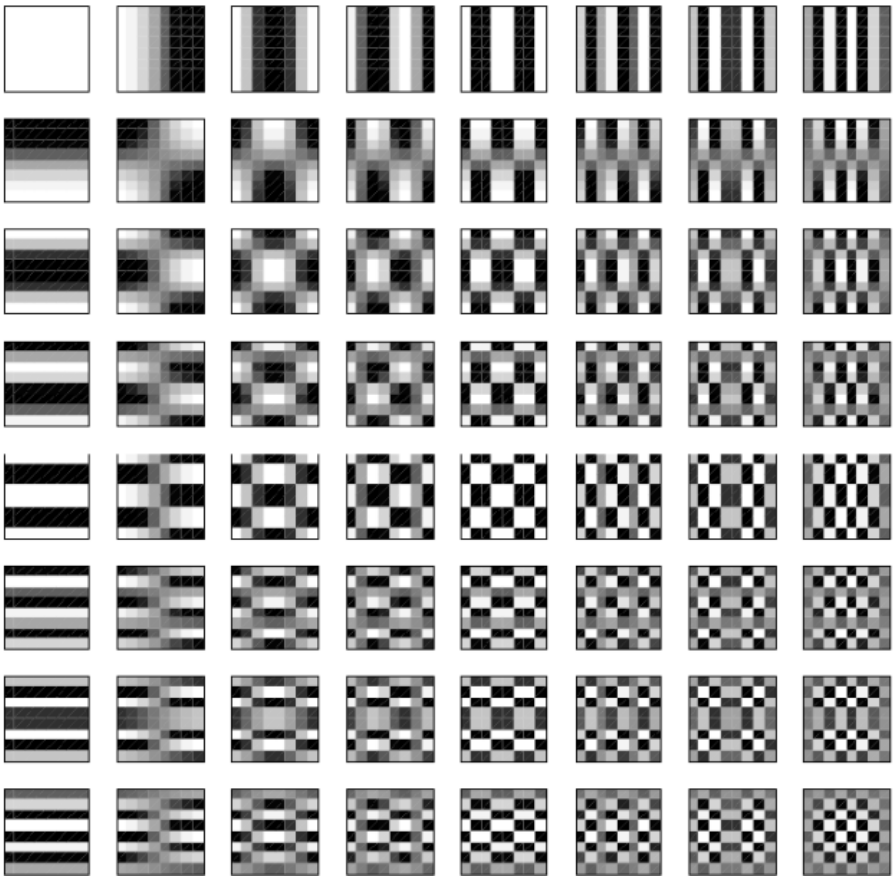

Рис. 7. Матрица базисных функций ДКП.