МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
КАФЕДРА «ЭВМ и системы»

ОТЧЁТ
по лабораторной работе № 4
Нелинейные методы контрастирования

Листов **9**

Выполнил студент группы Э-56
Занько Л. С.

Проверил Дубицкий А. В.

Брест 2020

Цель работы: Выделение контурных признаков изображения.

Задание: Составить программу, выполняющую выделение контурных признаков изображения.

Необходимые характеристики: изображение хранится во внешнем файле; программа должна выводить исходное и результирующее изображения; возможность выбора уровня порогового ограничения; перед выделением контура происходит фильтрация изображения любым фильтром. возможность изменения порога при роботе с контурным препаратом.

Вариант 4 Оператор Собела

Код программы:

```
#include <stdlib.h>
#include <stdio.h>
#include <png.h>
#include <math.h>

int width, height;
png_byte color_type;
png_byte bit_depth;
png_bytep *row_pointers = NULL;
float main_koef = 0.5;


unsigned char ***bufMatrix;


void read_png_file() {
        FILE *fp = fopen("/home/leon/volvo.png", "rb");

        png_structp png = png_create_read_struct(
          PNG_LIBPNG_VER_STRING, NULL, NULL, NULL);
        if (!png) abort();

        png_infop info = png_create_info_struct(png);
```

```
if (!info) abort();

if (setjmp(png_jmpbuf(png))) abort();

png_init_io(png, fp);

png_read_info(png, info);

width = png_get_image_width(png, info);
height = png_get_image_height(png, info);
color_type = png_get_color_type(png, info);
bit_depth = png_get_bit_depth(png, info);

// Read any color_type into 8bit depth, RGBA
   format.
// See http://www.libpng.org/pub/png/libpng-manual
   .txt

if (bit_depth == 16)
        png_set_strip_16(png);

if (color_type == PNG_COLOR_TYPE_PALETTE)
        png_set_palette_to_rgb(png);

// PNG_COLOR_TYPE_GRAY_ALPHA is always 8 or 16bit
   depth.
if (color_type == PNG_COLOR_TYPE_GRAY && bit_depth
   < 8)
        png_set_expand_gray_1_2_4_to_8(png);

if (png_get_valid(png, info, PNG_INFO_tRNS))
        png_set_tRNS_to_alpha(png);
```

```
        // These color_type don't have an alpha channel
           then fill it with 0xff.
        if (color_type == PNG_COLOR_TYPE_RGB ||
                color_type == PNG_COLOR_TYPE_GRAY ||
                color_type == PNG_COLOR_TYPE_PALETTE)
                png_set_filler(png, 0xFF, PNG_FILLER_AFTER
                   );

        if (color_type == PNG_COLOR_TYPE_GRAY ||
                color_type == PNG_COLOR_TYPE_GRAY_ALPHA)
                png_set_gray_to_rgb(png);

        png_read_update_info(png, info);

        if (row_pointers) abort();

        row_pointers = (png_bytep*)malloc(sizeof(png_bytep
           ) * height);
        for (int y = 0; y < height; y++) {
                row_pointers[y] = (png_byte*)malloc(
                   png_get_rowbytes(png, info));
        }

        png_read_image(png, row_pointers);

        fclose(fp);

        png_destroy_read_struct(&png, &info, NULL);
}

void write_png_file() {
        FILE *fp = fopen("/home/leon/lab3.png", "wb");
        if (!fp) abort();
```

```
png_structp png = png_create_write_struct(
  PNG_LIBPNG_VER_STRING, NULL, NULL, NULL);
if (!png) abort();

png_infop info = png_create_info_struct(png);
if (!info) abort();

if (setjmp(png_jmpbuf(png))) abort();

png_init_io(png, fp);

// Output is 8bit depth, RGBA format.
png_set_IHDR(
        png,
        info,
        width, height,
        8,
        PNG_COLOR_TYPE_RGBA,
        PNG_INTERLACE_NONE,
        PNG_COMPRESSION_TYPE_DEFAULT,
        PNG_FILTER_TYPE_DEFAULT
);
png_write_info(png, info);

// To remove the alpha channel for
  PNG_COLOR_TYPE_RGB format,
// Use png_set_filler().
//png_set_filler(png, 0, PNG_FILLER_AFTER);

if (!row_pointers) abort();

png_write_image(png, row_pointers);
```

```
        png_write_end(png, NULL);

        for (int y = 0; y < height; y++) {
                free(row_pointers[y]);
        }
        free(row_pointers);

        fclose(fp);

        png_destroy_write_struct(&png, &info);
}



void initBufMatrix() {
        bufMatrix = (unsigned char***)malloc(sizeof(
          unsigned char**) * height);
        for (int y = 0; y < height; y++) {
                bufMatrix[y] = (unsigned char**)malloc(
                  sizeof(unsigned char*) * width);
                for (int k = 0; k < width; k++) {
                        bufMatrix[y][k] = (unsigned char*)
                          malloc(sizeof(unsigned char) *
                          4);
                }
        }
}

void getBufMatrix() {
        for (int y = 0; y < height; y++) {
                for (int x = 0; x < width; x++) {
                        for (int k = 0; k < 4; k++) {
```

```
                                        bufMatrix[y][x][k] =
                                            row_pointers[y][x * 4 +
                                            k];
                                }
                        }
                }
        }

void delBufMatrix() {
        for (int y = 0; y < height; y++) {
                for (int x = 0; x < width; x++) {
                        free(bufMatrix[y][x]);
                }
                free(bufMatrix[y]);
        }
        free(bufMatrix);
}

void mainMath(int k, int y, int x) {
        unsigned int A, B, C, D, F, G, H, I;
        int t1, t2, Hh, Hv;
        float d;
        if (!(((y == (height - 1)) || (y == 0)) || ((x ==
            (width - 1)) || (x == 0)))) {
                A = bufMatrix[y - 1][x - 1][k];
                B = bufMatrix[y - 1][x][k];
                C = bufMatrix[y - 1][x + 1][k];
                D = bufMatrix[y][x - 1][k];
                F = bufMatrix[y][x + 1][k];
                G = bufMatrix[y + 1][x - 1][k];
                H = bufMatrix[y + 1][x][k];
                I = bufMatrix[y + 1][x + 1][k];
                t1 = A - I;
```

```
                t2 = C - G;
                Hh = 2 * (D - F) + t1 - t2;
                Hv = 2 * (B - H) + t1 + t2;
                float const_dr = main_koef;
                float buf_sum = pow((float)Hh, 2.0) + pow
                    ((float)Hv, 2.0);
                d = trunc(const_dr * pow(buf_sum, 0.5));
                row_pointers[y][x * 4 + k] = (unsigned
                    char)d;
        }
}

void process_png_file() {
        initBufMatrix();
        getBufMatrix();
        int k=0, x=0, y=0;
            for (k = 0; k < 3; k++) {
                    for (y = 0; y < height; y++) {
                            for (x = 0; x < width; x++) {
                                    mainMath(k, y, x);
                            }
                    }
            }
        delBufMatrix();
}
int main() {
        read_png_file();
        process_png_file();
        write_png_file();
        return 0;
}
```

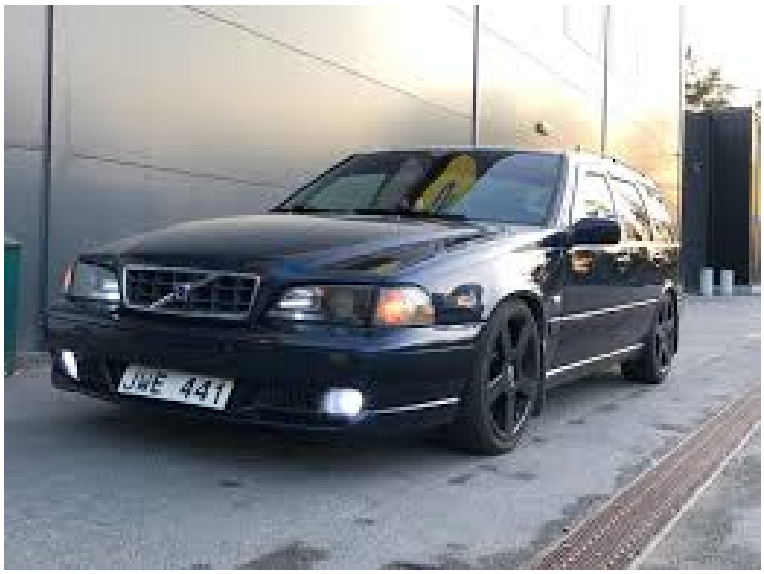

Рисунок 1 — Оригинальное изображение

Рисунок 2 — Полученное изображение

Вывод: В ходе работы изучил методы выделения контурных признаков изображения, составил программу выполняющую выделение контура методом Собела.