МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ВЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«Санкт-Петербургский Государственный университет Аэрокосмического Приборостроения»

Кафедра №14

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

доц.,к.т.н.		Н.В. Волошина
должность	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3
Графические фильтры
По курсу: «Основы мультимедия технологий».

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГРУППЫ 1441		А.А. Протасов
	подпись, дата	инициалы, фамилия

Санкт-Петербург
2016

1. Цель работы

Получение практических навыков реализации графических фильтров выполняющих преобразования изображений.

2. Постановка задачи

Для графических файлов в формате PNG реализовать фильтры выполняющие заданные преобразования изображения. Построить гистограммы для исходного и преобразованного изображений.
Графические файлы для проверки изображений создать самостоятельно.

3. Задание

Черно-белый, Фильтр яркость/контраст, Медианный фильтр

4. Краткие теоритические сведения

PNG - растровый формат хранения графической информации, использующий сжатие без потерь по алгоритму Deflate.
Deflate — это алгоритм сжатия без потерь, использующий комбинацию алгоритмов LZ77 и Хаффмана. Изначально был описан Филом Кацем для второй версии его архиватора PKZIP, который впоследствии был определён в RFC 1951[1] (1996 год).
Гистограмма изображения (иногда: график уровней или просто уровни) — гистограмма уровней насыщенности изображения (суммарная, или разделённая по цветовым каналам).

Гистограмма изображения позволяет оценить количество и разнообразие оттенков изображения, а также общий уровень яркости изображения.
Яркость — световая характеристика тел. Отношение силы света, излучаемого поверхностью, к площади ее проекции на плоскости, перпендикулярной оси наблюдения.
Контрастность — различимость предмета наблюдения от окружающего его фона (монохроматическое излучение); цветовая контрастность — разновидность оптической контрастности, связанная с разницей цветовых оттенков.

Иначе яроксть и контрастность можно описать так:
Яркость — количество белого цвета на вашем фото. Чем выше вы ставите яркость, тем светлее становится изображение, соответственно в обратную сторону оно будет темнее до черного.
Контрастность - разница между оттенками цвета предмета наблюдения и окружающего его фона. Если сформулировать проще то, это разница между различными расположенными рядом цветами. Чем выше контрастность, тем более резко мы наблюдаем переход от одного цвета к другому.

Контрастность была реализована по следующим формулам[2]:

$$C = (\frac{100.0+T}{100.0})^2 (1)$$

где C - коэффициент контрастности, а T- это значение от -100 до 100.

$$RGB = (((\frac{RGB^1}{255} - 0.5) \cdot C) + 0.5) \cdot 255 (2)$$

где RGB - компоненты красного, зеленого, синего пикселя, а C - это коэффициент из формулы (1)

Черно-белый фильтр - преобразование изображения в черно-белую гамму(каждый цветовой канал пикселя имеет одно значение).
Вычисляется по формуле[3] $\gamma = 0.299R + 0.587G + 0.144B$, где γ - новое значение компоненты, R, G, B - компоненты красного, зеленого и синего палитр.

Медианный фильтр - один из видов цифровых фильтров, широко используемый в цифровой обработке сигналов и изображений для уменьшения уровня шума. Медианный фильтр является нелинейным КИХ-фильтром.
Для упрощения дальнейшего рассмотрения ограничимся примером фильтра с квадратной маской размером $N * N$, при $N = 3$. Скользящий фильтр просматривает отсчеты изображения слева-направо и сверху-вниз, при этом входную двумерную последовательность также представим в виде последовательного числового ряда отсчетов $x(n)$ слева-направо сверху-вниз. Из этой последовательности в каждой текущей точке маска фильтра выделяет массив $w(n)$, как w-элементный вектор, который в данном случае содержит все элементы из окна $3*3$, центрированные вокруг $x(n)$, и сам центральный элемент, если это предусмотрено типом маски:

$$w(n) = [x_1(n), x_2(n), ..., x_W(n)](3)$$

В этом случае значения x_i соответствует отображению слева-направо и сверху-вниз окна $3 * 3$ в одномерный вектор, как показано на рис. 1.

$x_1(n)$	$x_2(n)$	$x_3(n)$
$x_4(n)$	$x(n)$	$x_5(n)$
$x_6(n)$	$x_7(n)$	$x_8(n)$

Рис. 1

Элементы данного вектора должны быть упорядочены в ряд по возрастанию или убыванию своих значений:

$$r(n) = [r_1(n), r_2(n), ..., r_W(n)](4)$$

Определено значение медианы $y(n) = med(r(n))$, центральный отсчет маски заменен значением медианы. Если по типу маски центральный отсчет не входит в число ряда (3), то медианное значение находится в виде среднего значения двух центральных отсчетов ряда (4). Приведенные выражения не объясняют способа нахождения выходного сигнала вблизи конечных и пограничных точек в конечных последовательностях и изображениях. Один из простых приемов состоит в том, что нужно находить медиану только тех точек внутри изображения, которые попадают в пределы апертуры. Поэтому для точек, расположенных рядом с границами, медианы будут определены, исходя из меньшего числа точек.

5. Описание метода реализации

Программа была написана в среде для разработки Qt[4] с использованием языка C++[5], фреймворка Qt[6] версии 5.7 и использованием библиотеки компьютерного зрения OpenCV[7] версии 2.4.13.1.

Для работы с изображениями были использованы следующие виды избражений:
32-бита(класс QImage - Qt) и 8-бит(класс Mat - OpenCV).
Для открытия изображения была использована стандартная функция imread() из библиотеки OpenCV. Так же для хранения изображения помимо типа Mat - библиотека OpenCV, был испльзован тип QImage - библиотека Qt, для взаимодействия между двумя разными способами хранинния избражений, были написаны две функции конвертирования QImageToMat - из представления Qt в OpenCV и MatToQImage - соответственно наоборот. Для разделения изображения на разные цветовые палитры был использован vector[8] - контейнер из STL и стандартная функция split из библиотеки OpenCV.

6. Листинги

6.1. Тело основной функции

```
#include "mainwindow.h"
#include "histogram.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    //w.show();
    w.QMainWindow::show();
    return a.exec();
}
```

6.2. Заголовок класса главного окна

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include "histogram.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private slots:
    void on_horizontalSlider_valueChanged(int value);

    void on_pushButton_pressed();

    void on_pushButton_2_pressed();

    void on_horizontalSlider_2_valueChanged(int value);

private:
    Ui::MainWindow *ui;
    Histogram hist;

};

inline QImage MatToQImage(const Mat &input);
```

```
35 inline Mat QImageToMat(const QImage &input);
36
37 #endif // MAINWINDOW_H
```

6.3. Описание методов класса главного окна

```
1  #include "mainwindow.h"
2  #include "ui_mainwindow.h"
3
4  Mat image = imread("/home/toshiki/files/картинки/PNG for LS/kek.png");
5
6  inline QImage MatToQImage(const Mat &input)
7  {
8      QImage image(input.data,
9                   input.cols, input.rows,
10                  static_cast<int>(input.step),
11                  QImage::Format_RGB888);
12     return image.rgbSwapped();
13 }
14
15 QImage img = MatToQImage(image);
16
17 inline Mat QImageToMat(const QImage &input)
18 {
19     QImage   swapped;
20     if(input.format() == QImage::Format_RGB32 )
21         swapped = input.convertToFormat( QImage::Format_RGB888 );
22     swapped = input.rgbSwapped();
23     return cv::Mat(swapped.height(), swapped.width(),
24                    CV_8UC3,
25                    const_cast<uchar*>(swapped.bits()),
26                    static_cast<size_t>(swapped.bytesPerLine())
27                   ).clone();
28 }
29
30 MainWindow::MainWindow(QWidget *parent) :
31     QMainWindow(parent),
32     ui(new Ui::MainWindow)
33 {
34     ui->setupUi(this);
35
36     ui->verticalLayout->setSpacing(0);
37     ui->verticalLayout->setMargin(0);
38     connect(ui->horizontalSlider, SIGNAL(valueChanged(int)),
39             this, SLOT(on_horizontalSlider_valueChanged(int)));
40     ui->label->setPixmap(QPixmap::fromImage(img).scaled(
41                              img.width()/2, img.height()/2, Qt::KeepAspectRatio));
42     ui->label->setMinimumWidth(img.width()/2);
43     ui->label->setMinimumHeight(img.height()/2);
44 }
45
46 inline int overflow(int value){
47     return (uint32_t)value>255?255:value<0?0:value;
```

```
48  }
49
50  QImage brightness_contrast(QImage &input, double alpha, int beta){
51      QImage temp = input;
52      qint32      height  = input.height();
53      qint32      width   = input.width();
54      for(int y = 0; y < height; ++y){
55          for(int x = 0; x < width; ++x){
56              QColor pix = temp.pixel(x,y);
57              int r = overflow(pix.red()+beta);
58              int g = overflow(pix.green()+beta);
59              int b = overflow(pix.blue()+beta);
60              int a = pix.alpha();
61              QRgb rgb = qRgba(r, g, b, a);
62              temp.setPixel(x,y,rgb);
63          }
64      }
65      for(int y = 0; y < height; ++y){
66          for(int x = 0; x < width; ++x){
67              QColor pix = temp.pixel(x,y);
68              int r = overflow(((pix.red()/255 - 0.5)*alpha + 0.5)*255);
69              int g = overflow(((pix.green()/255 - 0.5)*alpha + 0.5)*255);
70              int b = overflow(((pix.blue()/255 - 0.5)*alpha + 0.5)*255);
71              int a = pix.alpha();
72              QRgb rgb = qRgba(r, g, b, a);
73              temp.setPixel(x,y,rgb);
74          }
75      }
76      return temp;
77  }
78
79  MainWindow::~MainWindow()
80  {
81      delete ui;
82  }
83
84  void MainWindow::on_horizontalSlider_2_valueChanged(int value)
85  {
86      double alpha = (100.0 + value) / 100;
87      alpha = alpha * alpha;
88      ui->label_3->setText(QString::number(value));
89      QImage temp(brightness_contrast(
90                      img, alpha, 0).scaled(
91                          img.width()/2, img.height()/2,Qt::KeepAspectRatio));
92      ui->label->setPixmap(QPixmap::fromImage(temp));
93      Mat tmp = QImageToMat(temp);
94      std::vector<Mat> rgb_planes;
95      split(tmp, rgb_planes);
96      if(ui->checkBox->checkState() == Qt::Checked){
97          hist.show();
98          hist.histDisplay(rgb_planes[2], RED);
99          hist.histDisplay(rgb_planes[1], GREEN);
100         hist.histDisplay(rgb_planes[0], BLUE);
101     } else {
```

```
102         hist.hide();
103     }
104 }
105
106 void MainWindow::on_horizontalSlider_valueChanged(int value)
107 {
108     int beta = value * 2.55;
109     ui->label_2->setText(QString::number(value)+"%");
110     QImage temp(brightness_contrast(img, 1, beta).scaled(
111                     img.width()/2, img.height()/2,Qt::KeepAspectRatio));
112     ui->label->setPixmap(QPixmap::fromImage(temp));
113     Mat tmp = QImageToMat(temp);
114     std::vector<Mat> rgb_planes;
115     split(tmp, rgb_planes);
116     if(ui->checkBox->checkState() == Qt::Checked){
117         hist.show();
118         hist.histDisplay(rgb_planes[2], RED);
119         hist.histDisplay(rgb_planes[1], GREEN);
120         hist.histDisplay(rgb_planes[0], BLUE);
121     } else {
122         hist.hide();
123     }
124 }
125
126 void MainWindow::on_pushButton_pressed()
127 {
128     QImage input = ui->label->pixmap()->toImage();
129     qint32      height  = input.height();
130     qint32      width   = input.width();
131     for(qint32 y = 0; y < height; ++y){
132         QRgb* tempLine = reinterpret_cast<QRgb*>(input.scanLine(y));
133         for(qint32 x = 0; x < width; ++x){
134             int alpha = qAlpha(*tempLine);
135             int GS = qRed(*tempLine)   * 0.299 +
136                      qGreen(*tempLine) * 0.587 +
137                      qBlue(*tempLine)  * 0.114;
138             *tempLine++ = qRgba(GS, GS, GS, alpha);
139         }
140     }
141     if(ui->checkBox->checkState() == Qt::Checked){
142         hist.show(); hist.histDisplay(QImageToMat(input), GRAYSCALE);
143     }
144     ui->label->setPixmap(QPixmap::fromImage(input).scaled(
145                              img.width()/2, img.height()/2, Qt::KeepAspectRatio));
146 }
147
148 void MainWindow::on_pushButton_2_pressed()
149 {
150     QImage input = ui->label->pixmap()->toImage();
151     QVector<QRgb> vec;
152     for(int h = 1; h < input.height()-1; h++)
153         for(int w = 1; w < input.width()-1; w++){
154             vec.append(input.pixel(w-1,h-1));
155             vec.append(input.pixel(w,h-1));
```

```
156             vec.append(input.pixel(w+1,h-1));
157             vec.append(input.pixel(w-1,h));
158             vec.append(input.pixel(w,h));
159             vec.append(input.pixel(w+1,h));
160             vec.append(input.pixel(w-1,h+1));
161             vec.append(input.pixel(w,h+1));
162             vec.append(input.pixel(w+1,h+1));
163 
164             qSort(vec.begin(),vec.end());
165             input.setPixel(w,h,vec.at(4));
166             vec.clear();
167         }
168     Mat tmp = QImageToMat(input);
169     std::vector<Mat> rgb_planes;
170     split(tmp, rgb_planes);
171     if(ui->checkBox->checkState() == Qt::Checked){
172         hist.show();
173         hist.histDisplay(rgb_planes[2], RED);
174         hist.histDisplay(rgb_planes[1], GREEN);
175         hist.histDisplay(rgb_planes[0], BLUE);
176     } else {
177         hist.hide();
178     }
179     ui->label->setPixmap(QPixmap::fromImage(input).scaled(
180                              img.width()/2, img.height()/2, Qt::KeepAspectRatio));
181 }
```

6.4. Заголовок окна гистограммы

```
1  #ifndef HISTOGRAM_H
2  #define HISTOGRAM_H
3 
4  #include <QDialog>
5  #include <opencv2/highgui/highgui.hpp>
6  #include <opencv2/imgproc/imgproc.hpp>
7  #include <QtCharts>
8 
9  QT_CHARTS_USE_NAMESPACE
10 
11 using namespace cv;
12 
13 typedef enum{
14     RED,
15     GREEN,
16     BLUE,
17     GRAYSCALE
18 }plane;
19 
20 namespace Ui {
21 class Histogram;
22 }
23 
24 class Histogram : public QDialog
```

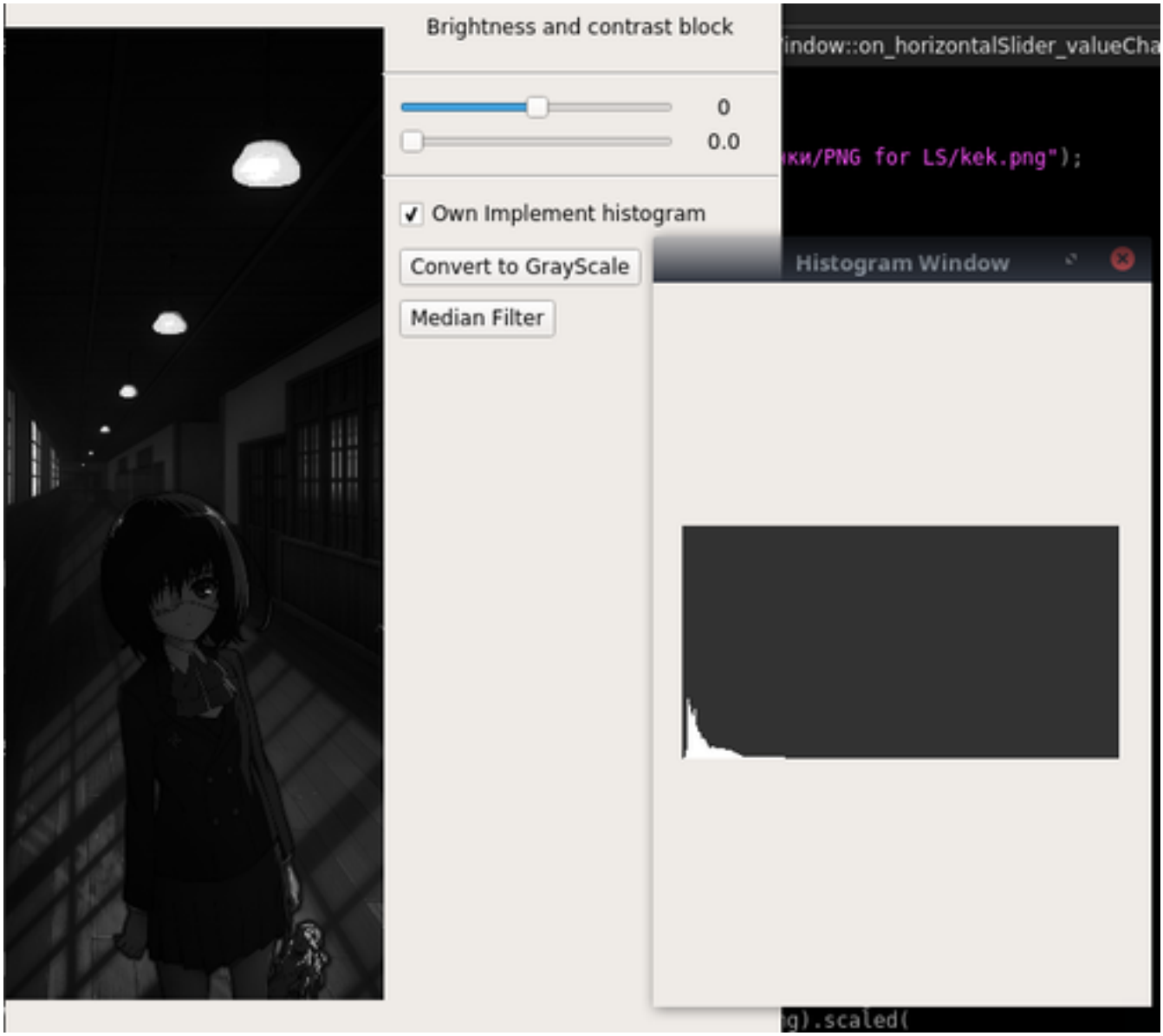
```
25 {
26     Q_OBJECT
27
28 public:
29     explicit Histogram(QWidget *parent = 0);
30     ~Histogram();
31     void histDisplay(Mat image, plane RGB);
32
33 private:
34     Ui::Histogram *ui;
35 };
36
37 #endif // HISTOGRAM_H
```

6.5. Описание методов класса окна гистограммы

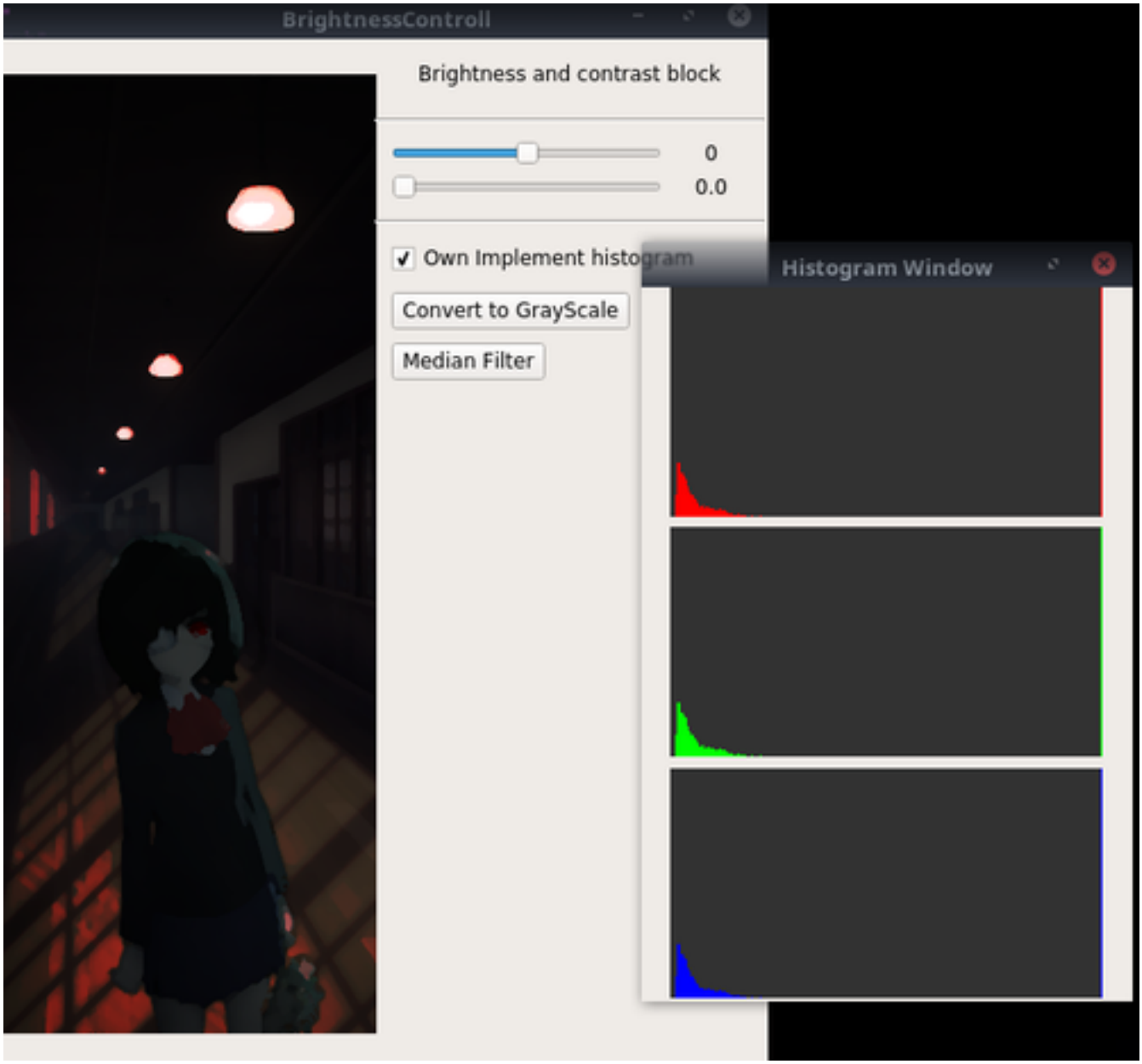
```
1  #include "histogram.h"
2  #include "ui_histogram.h"
3
4  inline QImage MatToQImage(const Mat &input)
5  {
6      QImage image(input.data,
7                   input.cols, input.rows,
8                   static_cast<int>(input.step),
9                   QImage::Format_RGB888);
10     return image.rgbSwapped();
11 }
12
13 Histogram::Histogram(QWidget *parent) :
14     QDialog(parent),
15     ui(new Ui::Histogram)
16 {
17     ui->setupUi(this);
18 }
19
20 Histogram::~Histogram()
21 {
22     delete ui;
23 }
24
25 void Histogram::histDisplay(Mat image, plane RGB)
26 {
27     int hist[256];
28     memset(hist, 0, 256*sizeof(*hist));
29     for(int y = 0; y < image.rows; y++)
30         for(int x = 0; x < image.cols; x++)
31             hist[(int)image.at<uchar>(y,x)]++;
32     int hist_w = 256;
33     int hist_h = ui->label_3->size().height();
34     int bin_w = cvRound((double) hist_w/256);
35     Mat histImage(hist_h, hist_w, CV_8UC3, Scalar(50, 50, 50));
36     int max = hist[0];
37     for(int i = 1; i < 256; i++)
```

```
38          if(max < hist[i])
39              max = hist[i];
40      for(int i = 0; i < 256; i++)
41          hist[i] = ((double)hist[i]/max)*histImage.rows;
42      switch ( RGB ) {
43      case RED:
44          for(int i = 0; i < 256; i++)
45                  line(histImage, Point(bin_w*(i), hist_h),
46                                          Point(bin_w*(i), hist_h - hist[i]),
47                       Scalar(0, 0, 255));
48          ui->label_3->setPixmap(QPixmap::fromImage(MatToQImage(histImage)));
49          ui->label_3->show();
50          break;
51      case GREEN:
52          for(int i = 0; i < 256; i++)
53                  line(histImage, Point(bin_w*(i), hist_h),
54                                          Point(bin_w*(i), hist_h - hist[i]),
55                       Scalar(0, 255, 0));
56          ui->label_4->setPixmap(QPixmap::fromImage(MatToQImage(histImage)));
57          ui->label_4->show();
58          break;
59      case BLUE:
60          for(int i = 0; i < 256; i++)
61                  line(histImage, Point(bin_w*(i), hist_h),
62                                          Point(bin_w*(i), hist_h - hist[i]),
63                       Scalar(255, 0, 0));
64          ui->label_5->setPixmap(QPixmap::fromImage(MatToQImage(histImage)));
65          ui->label_5->show();
66          break;
67      case GRAYSCALE:
68          for(int i = 0; i < 256; i++)
69                  line(histImage, Point(bin_w*(i), hist_h),
70                                          Point(bin_w*(i), hist_h - hist[i]),
71                       Scalar(255, 255, 255));
72          ui->label_3->setPixmap(QPixmap::fromImage(MatToQImage(histImage)));
73          ui->label_3->show();
74          ui->label_4->hide();
75          ui->label_5->hide();
76          break;
77      default:
78          break;
79      }
80  }
```

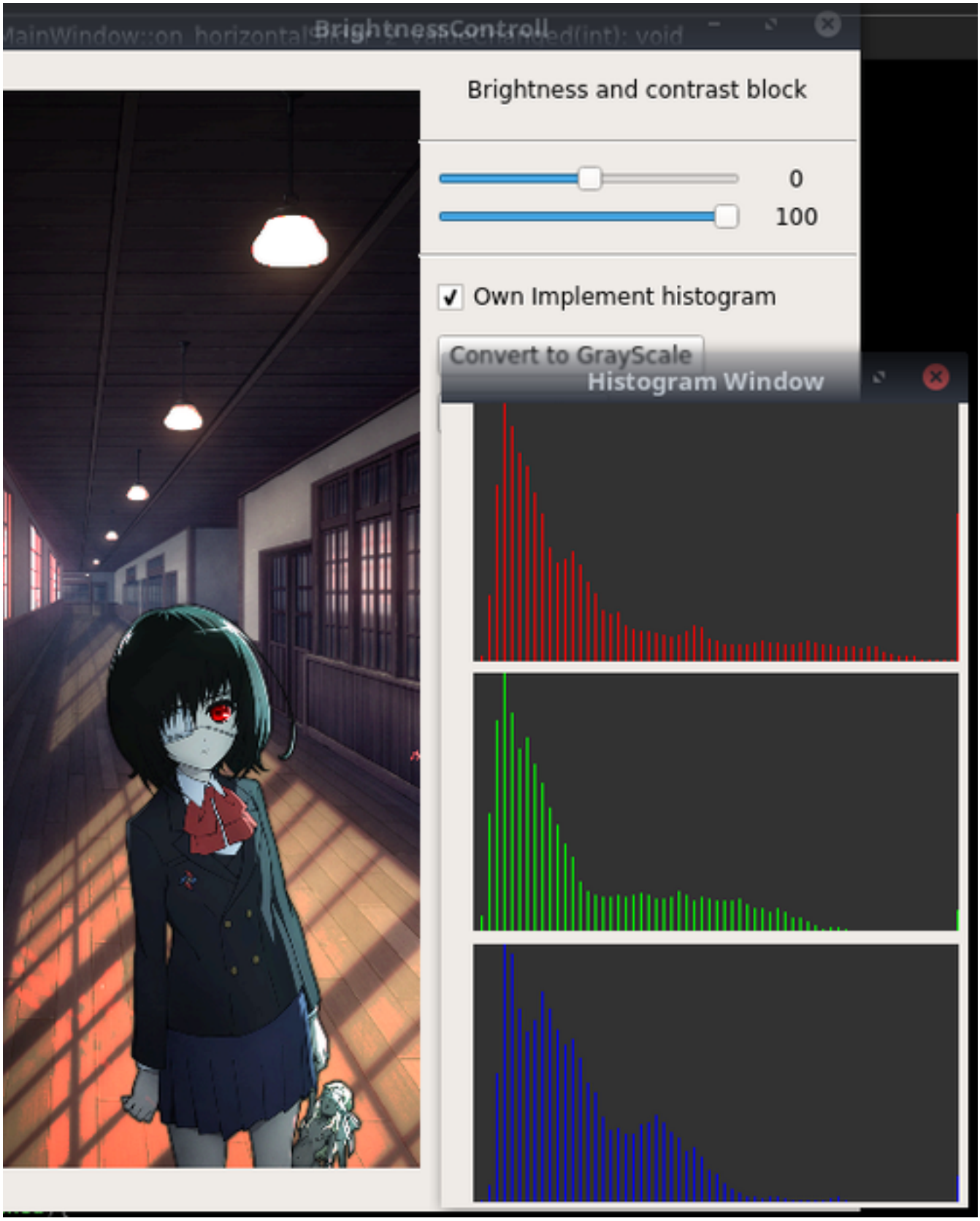
7. Примеры работы программы



Черно-белый


BrightnessControll
Brightness and contrast block
0
0.0
Own Implement histogram
Convert to GrayScale
Median Filter
Histogram Window

Медианный



Контрастность

Список литературы

[1] https://tools.ietf.org/html/rfc1951ref-1

[2] https://softwarebydefault.com/2013/04/20/image-contrast/

[3] https://ru.wikipedia.org/wiki/Оттенки_серого

[4] Шлее М. Qt 5.3. Профессиональное программирование на C++ — Санкт-Петербург: Изд. БХВ-Петербург, 2015. 928 с.

[5] http://en.cppreference.com/w/

[6] http://doc.qt.io/qt-5/reference-overview.html

[7] http://opencv.org/

[8] http://en.cppreference.com/w/cpp/container/vector