**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ**

***Институт Принтмедиа и информационных технологий***

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»,**

**ЛАБОРАТОРНАЯ РАБОТА № 3**

**Дисциплина:** Технология кроссплатформенного программирования

**Тема:** Редактор фото

**Выполнил: студент группы ДЦисБ-2-2**

**Матевосян В.А.**

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Проверил: Северинов Н.А.**

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Замечания:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва**

**2017**

**Содержание**

1. Теоретическое обоснование…………………………………...3
2. Листинг программы……………………………………………5
3. Скриншоты программы………………………………………..8

**Теоретическое обоснование**

**Фильтрация изображений** аналогична такому разглядыванию мира через стеклянную пластину, хотя и позволяет добиться гораздо большего разнообразия эффектов, чем эксперименты с разными пластинами. Под фильтрацией изображений понимают операцию, имеющую своим результатом изображение того же размера, полученное из исходного по некоторым правилам. Обычно интенсивность (цвет) каждого пикселя результирующего изображения обусловлена интенсивностями (цветами) пикселей, расположенных в некоторой его окрестности в исходном изображении.

Правила, задающие фильтрацию (их называют **фильтрами** ), могут быть самыми разнообразными. В этой лекции мы рассмотрим простейшие фильтры. Заметим, что, согласно предложенному определению, операция, заключающаяся в последовательном применении двух или более фильтраций, тоже является фильтрацией. Таким образом, можно говорить о **составных фильтрах**, соответствующих комбинациям простых. Изучив основные типы фильтров в данной лекции, мы будем иметь дело с применением фильтров, составленных из них, для решения разнообразных задач в последующих лекциях. Фильтрация изображений является одной из самых фундаментальных операций компьютерного зрения, распознавания образов и обработки изображений. Фактически, с той или иной фильтрации исходных изображений начинается работа подавляющего большинства методов. Рассматриваемые в этой лекции фильтры имеют, таким образом, чрезвычайную важность с точки зрения их применения в различных приложениях.

Говоря не математическим языком, **convolution** – это преобразование одной матрицы с помощью другой, которая называется *ядром* («kernel»). При обработке изображений в качестве исходных выступают матрицы RGB-каналов пикселей в прямоугольных координатах.   
В качестве ядра обычно используется матрица размером 3x3, но возможно и больше (5x5, 7x7 и т.д.). Ядро содержит *степени влияния («ценности»)* окружающих значений элемента на сам элемент.   
  
Преобразования происходит следующим образом. Каждый элемент исходной матрицы умножается центральное значение матрицы ядра. Кроме этого на соответствующие значения умножаются окружающие его элементы (при размере ядра 3x3 их будет 8), после чего результаты суммируются и принимаются как преобразованное значение.

Повышение/снижение яркости – это, соответственно, сложение/вычитание значения каждого канала с некоторым фиксированным значением (также в пределах от 0 до 255); при этом **обязательно** необходимо контролировать выход нового значения канала за пределы диапазона 0..255.

Повышение/снижение контрастности – это, соответственно, умножение/деление значения каждого канала на некоторое фиксированное значение (в том числе действительное), что приводит к изменению соотношений между цветами и, соответственно, к более чётким цветовым границам. На практике же существует такой принцип: изменение контрастности не должно приводить к изменению средней яркости по изображению, поэтому пользуются следующей формулой:

*NewY := K \* (OldY - AvgY) + AvgY,*

где *NewY* – новое значение одного из каналов, *K* – коэффициент контрастности (*K=(0..1)* – снижение, *K>1* – повышение контрастности), *OldY* – текущее значение того же канала, *AvgY* – среднее значение того же канала по изображению (таким образом, алгоритм фактически является двухпроходовым). **Обязательна** всё та же коррекция нового значения при выходе его за границы 0..255.

Под *изменением цветности* понимается изменение спектра цветов, используемых в изображении. Минимальное, что нужно сделать в курсовой работе, – бинаризация, оттенки серого и негатив, поэтому о них и поговорим.

**Бинаризация**

Это преобразование изображения, в общем случае, к одноцветному (чаще всего к чёрно-белому). В терминах Photoshop это ещё называется «по уровню 50%», так как при этом выбирается некий порог (например, посередине), все значения ниже которого превращаются в цвет фона, а выше – в основной цвет. Само преобразование можно осуществлять по каналам, но в этом случае результирующее изображение не будет в прямом смысле бинарным (чёрно-белым), а будет содержать 8 чистых цветов, представляющих собой комбинации чистых красного, зелёного и голубого цветов, то есть будет бинарным по каналам.

**Преобразование к оттенкам серого**

Заключается в получении яркости каждой точки по известной формуле (*Y := 0.3 \* R + 0.59 \* G + 0.11 \* B*) и последующем копировании полученного значения во все три канала (*R = G = B := Y*).

**Листинг программы**

Листинг widget.h

#ifndef WIDGET\_H

#define WIDGET\_H

#include <QWidget>

namespace Ui {

class Widget;

}

class Widget : public QWidget

{

Q\_OBJECT

public:

explicit Widget(QWidget \*parent = 0);

~***Widget***();

private slots:

void **on\_pushButton\_clicked**();

void **on\_pushButton\_2\_clicked**();

void **on\_pushButton\_3\_clicked**();

private:

Ui::Widget \*ui;

};

#endif // WIDGET\_H

#include "widget.h"

Листинг main.cpp

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

Widget w;

w.show();

return a.exec();

}

#include "widget.h"

Листинг widget.cpp

#include "ui\_widget.h"

#include <QFileDialog>

#include <QImage>

#include <QPixmap>

#include <QGraphicsScene>

Widget::**Widget**(QWidget \*parent) :

QWidget(parent),

ui(new Ui::Widget)

{

ui->setupUi(this);

ui ->label ->resize(640, 380 );

}

Widget::~***Widget***()

{

delete ui;

}

QString a;

QRgb p;

void Widget::**on\_pushButton\_clicked**()

{

a=QFileDialog::getOpenFileName();

QImage image(a);

ui->label->setPixmap ( QPixmap::fromImage(image).scaled( 640, 380 ) );

}

void Widget::**on\_pushButton\_2\_clicked**()

{

QImage image (a);

ui->label->setPixmap( QPixmap::fromImage(image).scaled( 640, 380 ) );

}

void Widget::**on\_pushButton\_3\_clicked**()

{

QImage image(a);

QImage imagegray(a);

ui->label->setPixmap( QPixmap::fromImage(image).scaled( 640, 380 ) );

for ( int x=0; x<image.width (); x++ )

{

for ( int y=0; y<image.height(); y++ )

{

p=image.pixel( x, y);

imagegray.setPixel ( x, y, qRgb ( qRed(p)/3 + qGreen(p)/3 + qBlue(p)/3, qRed(p)/3 + qGreen(p)/3 + qBlue(p)/3, qRed(p)/3 + qGreen(p)/3 + qBlue(p)/3 ) );

}

}

ui->label->setPixmap ( QPixmap::fromImage(imagegray).scaled( 640, 380 ) );

}

**Скриншоты программы**

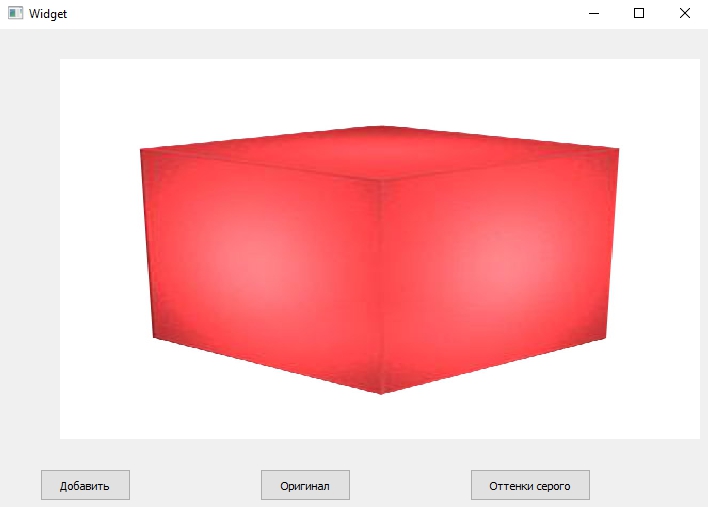


Рисунок (1) Добавление фото-оригинала.

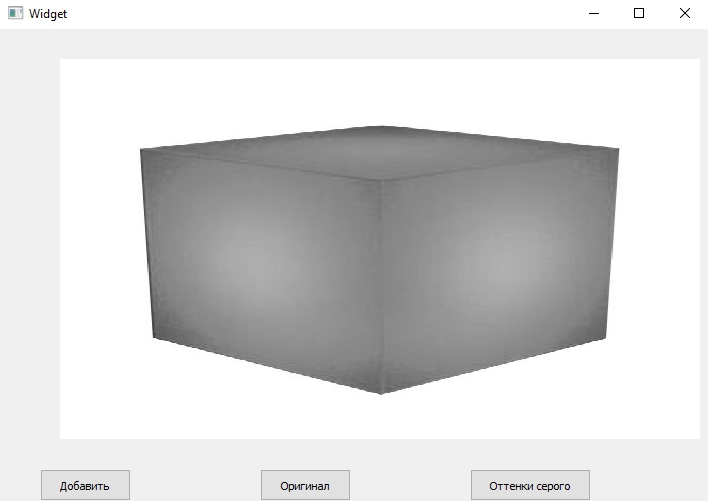


Рисунок (2) Использование фильтра “Оттенки серого”