Министерство образования и науки Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Физический факультет
Кафедра астрономии и космической геодезии

ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК

Руководитель ООП
д-р физ.-мат. наук
_____ В.А. Авдюшев
«___» _____ 2019 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ОБРАБОТКИ ЦИФРОВЫХ ИЗОБРАЖЕНИЙ KOENIGSBERG 1.1

по основной образовательной программе подготовки бакалавров
направление подготовки 09.03.02 – Информационные системы и технологии

Шмидт Екатерина Евгеньевна

Руководитель
к. ф.-м. н., доцент кАиКГ ФФ
_____ М.А. Баньщикова
«___» _____ 2019 г.

Автор работы
Студент группы №557б
_____ Е.Е. Шмидт

Томск-2019

МИНОБРНАУКИ РОССИИ
Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский
Томский государственный университет»
Физический факультет
Кафедра астрономии и космической геодезии

УТВЕРЖДАЮ
Руководитель ООП
д-р физ.-мат. наук
Авдюшев В. А. _____
14 сентября 2018 г.

ЗАДАНИЕ
по подготовке ВКР бакалавра Шмидт Е.Е. группы № 557б

1. Тема ВКР «Программное обеспечение для обработки цифровых изображений Koenigsberg 1.1».
2. Срок сдачи студентом выполненной ВКР: июнь 2019 г.
3. Исходные данные к работе: теория по режимам и фильтрам, применяемых в обработке цифровых изображений.
4. Краткое содержание работы:
   1 глава — алгоритмы обработки цифровых изображений, используемых в ПО, 2 глава — разработка ПО Koenigsberg 1.1, 3 глава — сравнительный анализ одноименных эффектов/фильтров Koeinigsberg 1.1 с GIMP 2.10.10.
5. Организации, по заданию которых выполнялась работа:
   Томский государственный университет (ТГУ).
6. Графический материал: 45 рисунков.
7. Дата выдачи задания: 14 сентября 2018 г.

Руководитель ВКР,
к. физ.-мат. наук _____ Баньщикова М.А.

Задание принял к исполнению
студент группы № 557б _____ Шмидт Е.Е.

РЕФЕРАТ

Выпускная квалификационная работа «Программное обеспечение для обработки цифровых изображений Koenigsberg 1.1» содержит 45 страниц печатного текста, 45 рисунков, 3 формулы и 11 источников использованной литературы.

Ключевые слова: пиксел, режим градации серого, режим Black and White негатив, яркость, контрастность, линейное контрастирование, гистограмма, шум, медианный фильтр, сглаживающий фильтр.

Цель работы – создание программного обеспечения для обработки цифровых изображений.

В процессе разработки программного обеспечения использовались:

* Turbo Delphi 2006— среда разработки программного обеспечения.

* Язык программирования Delphi.

Результат работы — программное обеспечение для обработки цифровых изображений

Содержание

Введение

Как известно цифровая обработка изображений – интенсивно развивающаяся научная область, которая находит все более широкое применение в различных информационных технических системах: радиолокационных, связи, телевизионных, космических и др. Это связано с непрерывным увеличением количества аппаратных средств, являющихся источниками визуальной (зрительной) информации.

Ещё в середине XX века обработка изображений была по большей части аналоговой и выполнялась оптическими устройствами. Подобные оптические методы до сих пор важны, в таких областях как, например, голография. Тем не менее, с резким ростом производительности компьютеров, эти методы всё чаще стали вытесняться методами цифровой обработки изображений. Методы цифровой обработки изображений обычно являются более точными, надёжными, гибкими и простыми в реализации, нежели аналоговые методы. К сожалению, не один из методов/фильтром не является универсальным – требуются новые подходы.

Целью данной выпускной квалификационной работы является создания интегрированного простого и оперативного редактора космических изображений/снимков.

Необходимость разработки ПО заключается во внедрении в ПО Вектор-М [1] для обработки комических снимков.

Задачами работы является освоение и реализация алгоритмов по обработке цифровых изображений:

- Режим черно-белый;
- Негатив;
- Режим градации серого;
- Изменение яркости и контрастности;
- Построение гистограмм;
- Линейное контрастирование;
- Шум;
- Сглаживающий фильтр;
- Медианный фильтр.

1. Алгоритмы обработки цифровых изображений

1.1 Основные понятия

Растровое изображение представляет собой сетку из пикселов, каждый из которых имеет определенные горизонтальные и вертикальные координаты внутри сетки [2].

Пи́ксель, пи́ксел — наименьший логический элемент двумерного цифрового изображения в растровой графике. Пиксель представляет собой неделимый объект прямоугольной или круглой формы, характеризуемый определённым цветом [3].

Цветовые модели (color model) [4] используются для математического описания определенных цветовых областей спектра. Большинство компьютерных цветовых моделей основано на использовании трех основных цветов, что соответствует восприятию цвета человеческим глазом. Каждому основному цвету присваивается определенное значение цифрового кода, после чего все остальные цвета определяются как комбинации основных цветов.

В данной работе мы рассматриваем RGB цветовую модель, которая является аддитивной и, соответственно, основана на сложении цветов.

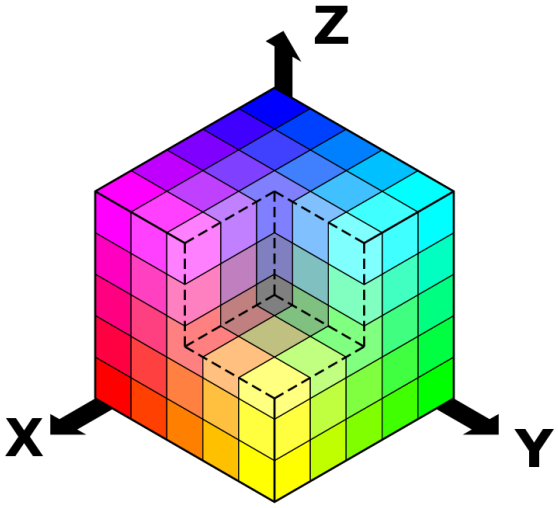

Рисунок 1.1. – Цветовая модель RGB

Цветовую модель *RGB* удобнее всего представлять в виде куба (Рисунок 1.1.). В этом случае каждая его пространственная точка однозначно определяется значениями координат X, Y и Z. Если по оси X откладывать красную составляющую, по оси Y − зеленую, а по оси Z − синюю, то каждому цвету можно поставить в соответствие точку внутри куба [2].

1.2. Режим Black and White

Изображение в режиме Black and White (Рисунок 1.2.) характеризуется высоким контрастом и отсутствием полутонов. Поэтому при конвертировании цветного изображения в данный режим, цвета окрашиваются либо в черный, либо в белый цвет [2].

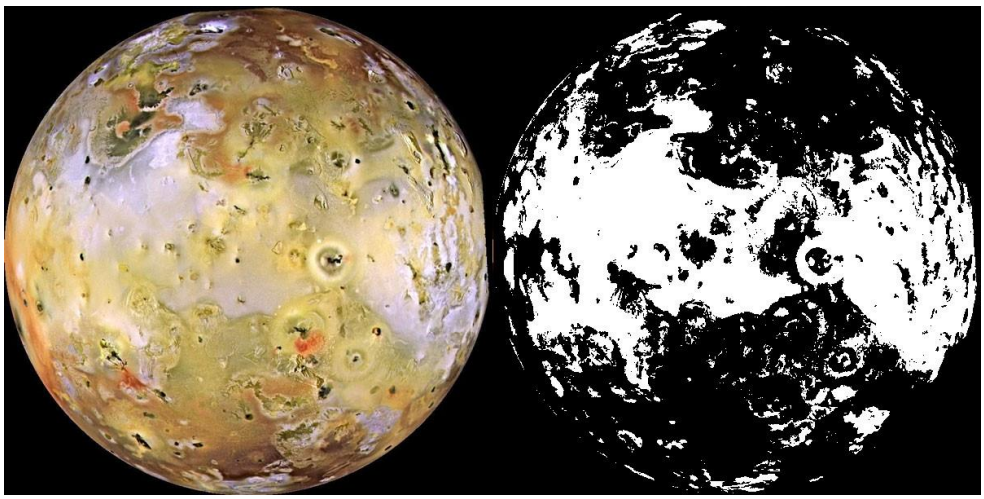

Рисунок 1.2. – Режим Black and White

Преобразование изображения можно осуществлять по каналам, но в этом случае результирующее изображение не будет в прямом смысле бинарным (чёрно-белым), а будет содержать восемь чистых цветов, представляющих собой комбинации чистых красного, зелёного и голубого цветов, то есть будет бинарным по каналам. Поэтому проведено преобразование над «полным» цветом пикселя, т.е. без разделения каналов.

Алгоритм, заключается в следующем: находится порог и в зависимости от этого значения пиксел p_i становится либо белым, либо черным (1):

$$MidColor = (Black - White) / 2 \quad \Rightarrow \quad \begin{matrix} \text{Если} & p_i < MidColor & p_i = White \\ \text{Иначе} & & p_i = Black, \end{matrix} \qquad (1)$$

где *MidColor* – порог.

Ниже представлен код, реализующий данный режим:

```
…
1.  MidColor :=  ($00FFFFFF-$00000000) div 2;
2.  for x:=1 to BitMap.Width-1 do
3.  begin
4.  for y:=1 to BitMap.Height-1 do
5.  begin
6.  Pixel := BitMap.Canvas.Pixels [x, y];
7.  if Pixel < MidColor
8.  then Pixel := $00000000                    //черный цвет
9.  else Pixel := $00FFFFFF;                   //белый цвет
10. BitMap.Canvas.Pixels [x, y] := Pixel;
11. end;
12. end;
    …
```

1.3. Негатив

Применимо к любому изображению, термин «негатив» (Рисунок 1.3.) означает, что тона и цвета такого изображения прямо противоположны цветам и тонам оригинала [5].

Негатив получается простой заменой значения каждого канала на его дополнение до 255, т.е. R=255-R; G=255-G и B=255-B.

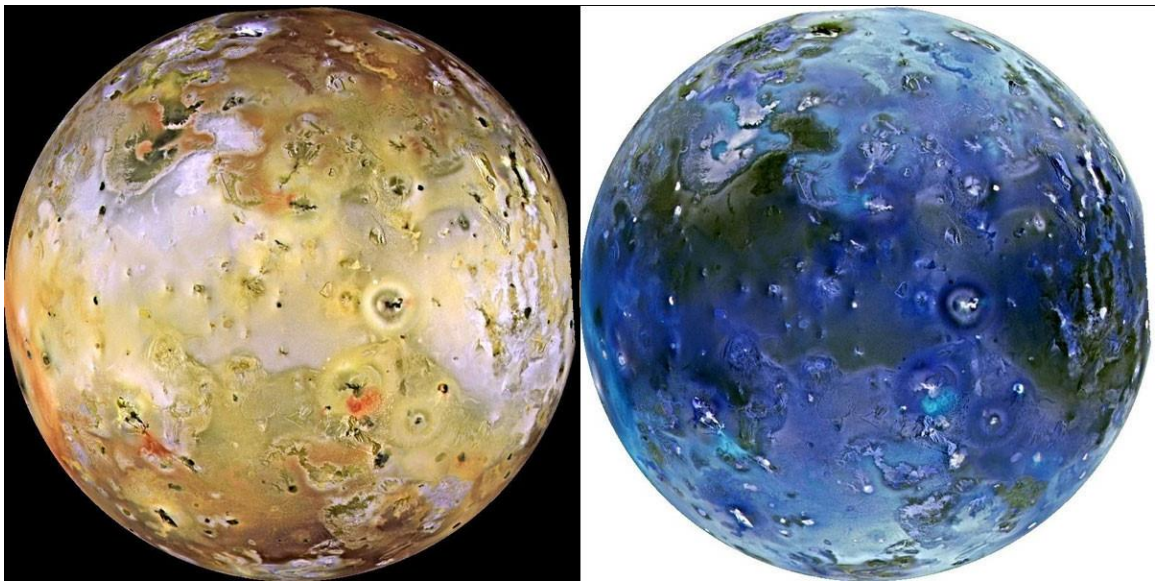

Рисунок 1.3. – Негатив

Ниже представлен код, реализующий данный режим:

```
…
rgb:=ColorToRgb(BitMap.Canvas.Pixels[x,y]);
r:=255-getRvalue(rgb);          //высчитывает соответствующую
g:=255-getGvalue(rgb);          //компоненту
b:=255-getBvalue(rgb);          //цвета
BitMap.Canvas.Pixels[x,y]:=windows.RGB(r,g,b);
…
```

1.4. Режим Grayscale

В режиме черно-белой графики мы оперировали только двумя цветами, тогда как в режиме Grayscale (Рисунок 1.4.) для пиксела с 4-битовым разрешением число возможных вариантов составит 2^4, что соответствует 16 комбинациям. В случае 8-битового разрешения это число возрастет до 28, или 256 комбинаций. Растровые редакторы воспринимают полученное в этом режиме цифровое изображение в виде одноцветного (монохромного) канала, содержащего 256 различных уровней яркости [2].

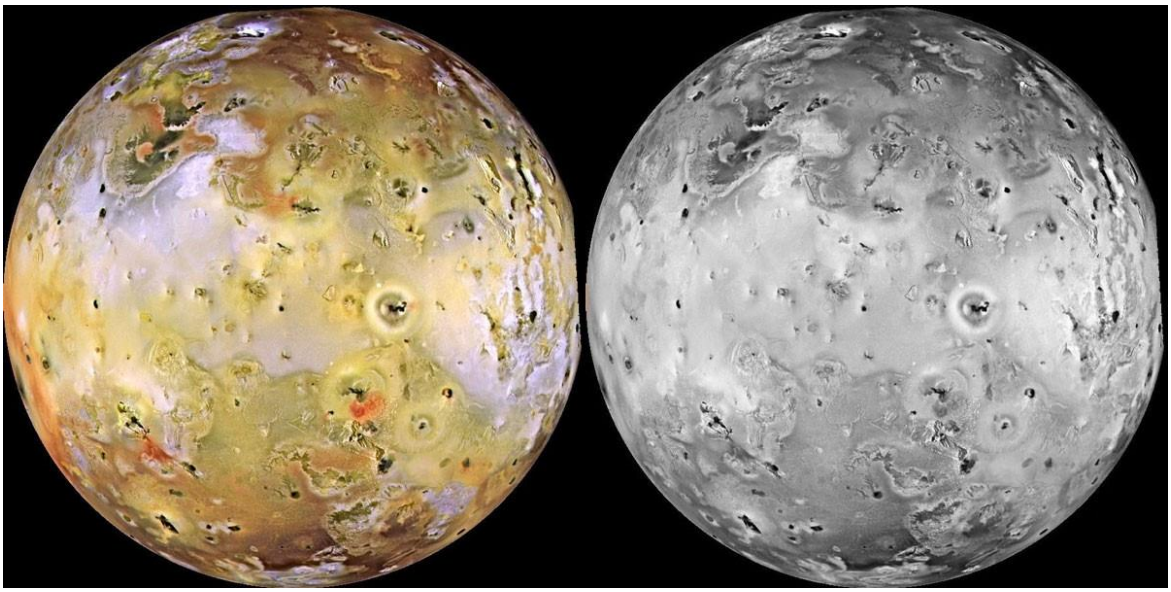

Рисунок 1.4. – Режим Grayscale

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 по все три канала (R=G=B=Y).

Ниже представлен код, реализующий данный режим:

```
1.  rgb:=ColorToRgb(BitMap.Canvas.Pixels[x,y]);
2.  r := getRvalue(rgb);                    //высчитывает соответствующую
3.  g := getGvalue(rgb);                    //компоненту
4.  b := getBvalue(rgb);                    //цвета
5.  gscale:=round((0.30*r)+(0.59*g)+(0.11*b));  //формула яркости каждой точки
6.  BitMap.Canvas.Pixels[x,y]:=windows.RGB(GScale,GScale, GScale);
```

1.5. Изменение яркости

Понятие яркости в цифровой графике ближе к обычному использованию этого слова: в общем случае под яркостью понимается преобладание тех или иных тонов.

При избытке светлых тонов говорят о значительной яркости (в цифровых технологиях она определяется как положительная яркость), при преобладании темных тонов говорят о незначительной яркости (в цифровых технологиях она определяется как отрицательная яркость).

Яркость изображения определяет общее впечатление от изображения. Поэтому на этот параметр обращают внимание в первую очередь. Для изображения важна, прежде всего, тоновая сбалансированность. Однако существуют причины, по которым необходимо уменьшить или увеличить яркость изображения [6].

Увеличение/уменьшение яркости – это, соответственно, сложение или 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.

Алгоритм заключается в следующем: необходимо найти фиксированное значение Δ, на которое будут отличаются компоненты цвета r_i, g_i, b_i (2):

$$\begin{aligned} &\gamma = 255 - min + max, \\ &\Delta = \sigma\ \left(\gamma\ /\ 2 + \gamma\ \sigma/200 \right)/100, \end{aligned} \quad \Rightarrow \quad \begin{aligned} &r_i, g_i, b_i = r_i, g_i, b_i + \Delta, \\ &p_i = r_i, g_i, b_i, \end{aligned} \qquad (2)$$

где p_i – пиксел картинки, r_i, g_i, b_i – компонента цвета пикселя p_i $(r_i, g_i, b_i = [0,255])$, σ – положение бегунка, min, max – минимум и максимум среди r_i, g_i, b_i, где $i = \overline{1,n}$, n – количество пикселей в изображении.

Ниже представлен код, реализующий изменение яркости:

```
…
1.  rgb:=ColorToRgb(BitMap.Canvas.Pixels[x,y]);
2.  r := getRvalue(rgb);                    //высчитывает соответствующую
3.  g := getGvalue(rgb);                    //компоненту
4.  b := getBvalue(rgb);                    //цвета
5.  mi:=Min(Min(r,g),b);                   //минимальное значение среди r,g,b
6.  ma:=Max(Max(r,g),b);                   //максимальное значение среди r,g,b
7.  plus:=Position*((255-mi+ma)/2+(255-mi-ma)*Position/200)/100;      //значение
    //для изменения яркости
8.  r:=Blimit(r+Trunc(plus));
9.  g:=Blimit(g+Trunc(plus));
10. b:=Blimit(b+Trunc(plus));
11. BitMap.Canvas.Pixels[x,y]:=windows.RGB(r,g, b);
…
```

Для определения фиксированного значения находится максимальное и минимальное значение среди каналов R, G, B. В строчке 7 кода написан алгоритм нахождения значения, необходимого для реализации изменения яркости. Функция Blimit проверяет выход значения канала за пределы диапазона.

1.6. Изменение контрастности

Контрастность – одна из основных характеристик изображения, напрямую связанная с яркостью пикселей.

При увеличении контрастности изображения светлые участки (пиксели) становятся еще светлее, а темные темнее. В результате происходит перераспределение пикселей за счет среднего тонового диапазона. Некоторые из них переходит в светлую часть, а некоторые – в тени.

При уменьшении контрастности изображения, наоборот происходит расширение среднего тонового диапазона за счет пограничных светов и

теней. Темные пиксели становятся более светлыми, а светлые более темными и частично переходят в средние тона [7].

Увеличение/уменьшение 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.

Ниже представлен код реализации изменения контрастности [8]:

```
…
if Local then  begin
mR := 128;
mG := 128;
mB := 128;  end
else  begin
tr := 0;
tg := 0;
tb := 0;
for y := 0 to H do    begin
Dest := BitMap.ScanLine[y];
for x := 0 to W do begin
with Dest^ do begin
Inc(tb, rgbtBlue);
Inc(tg, rgbtGreen);
Inc(tr, rgbtRed); end;
Inc(Dest); begin;
mB := Trunc(tb / (W * H));
mG := Trunc(tg / (W * H));
mR := Trunc(tr / (W * H)); end;
if Position > 0 then vd := 1 + (Position / 10)
else vd := 1 - (Sqrt(-Position) / 10);
for y := 0 to H do begin
Dest := Bitmap.ScanLine[y];
for x := 0 to W do begin
with Dest^ do begin
rgbtBlue := BLimit(mB + Trunc((rgbtBlue - mB) * vd));
rgbtGreen := BLimit(mG + Trunc((rgbtGreen - mG) * vd));
rgbtRed := BLimit(mR + Trunc((rgbtRed - mR) * vd)); end;
Inc(Dest); end;
…
```

1.7. Гистограммы

Гистограмма изображения (Рисунок 1.5.) (график уровней или просто уровни) – гистограмма уровней насыщенности изображения (суммарная, или разделённая по цветовым каналам) [9].

По оси абсцисса располагаются значения яркости от 0 до 255, то есть от самого темного (черного) до самого светлого (белого). По оси ордината количество пикселей той или иной яркости. Чем выше столбец, тем больше пикселей данной яркости на изображении.

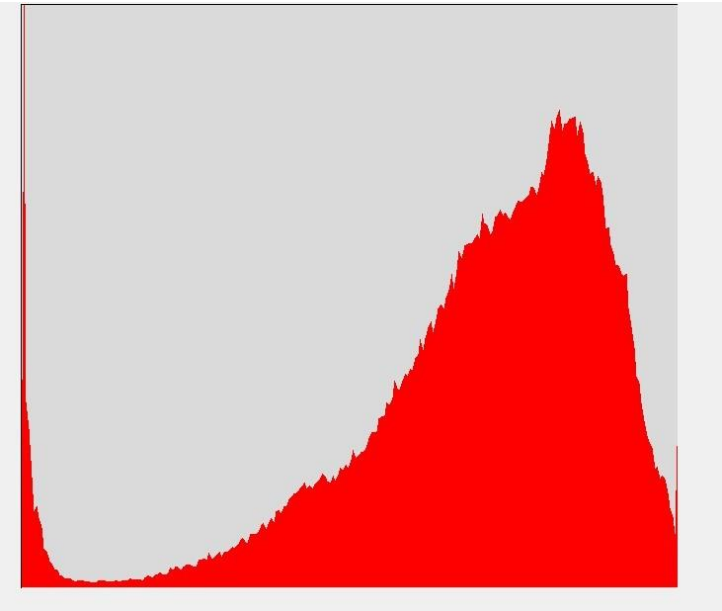

Рисунок 1.5. – Гистограмма

Ниже представлен код, реализующий построение гистограммы для красного канала (аналогичные действия производятся для зеленого и синего каналов):

```
1.  Series1.Clear;
2.  for i:=0 to 255 do Red[i]:=0;
3.  for j:=1 to FormMain.ImageMain.Picture.Bitmap.Height-1 do begin
4.  Row:=pRGBArray(FormMain.ImageMain.Picture.Bitmap.Scanline[j]);
5.  for i:=1 to FormMain.ImageMain.Picture.Bitmap.Width-1 do begin
6.  Red[Row[i].rgbtRed]:=Red[Row[i].rgbtRed]+1; end;
7.  for i:=0 to 255 do Series1.Add(Red[i],IntToStr(i),clRed);
```

1.8. Линейное контрастирование

Слабый контраст – распространенное свойство изображений, которое обусловлено ограничением диапазона воспроизводимых яркостей и др. Задача контрастирования связана с улучшением согласования динамического диапазона изображения и экрана, на котором выполняется визуализация (Рисунок 1.6.). Если для цифрового представления каждого отсчета изображения отводится 1 байт (8 бит) запоминающего устройства, то входной или выходной сигналы могут принимать одно из 256 значений. В качестве рабочего диапазона используются значения сигнала [0,255]; при этом значение 0 соответствует при визуализации уровню черного, а значение 255 – уровню белого. Предполагается, что минимальная и максимальная яркости исходного изображения равны *fmin* и *fmax* соответственно. Если хотя бы один из этих параметров существенно отличаются от граничных значений яркостного диапазона, то визуализированная картина выглядит либо как темная, либо как ненасыщенная, неудобная, утомляющая при наблюдении.

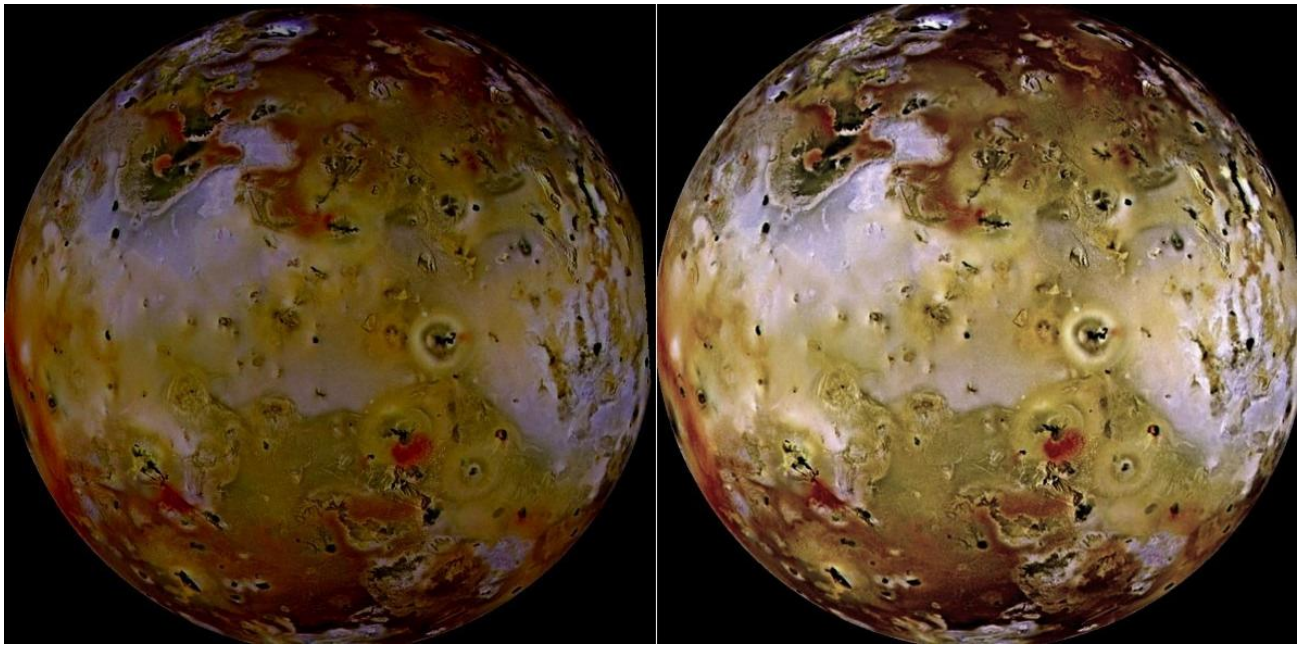

Рисунок 1.6. – Линейное контрастирование

Алгоритм заключается в следующем: необходимо найти фиксированное значение δ, на которое будут отличаются компоненты цвета r_i, g_i, b_i (3):

$$\begin{matrix} gg = (Y - Y_{min}) / (Y_{\max} - Y_{min})\Delta g + g_{min} \\ \delta = Y - gg, \end{matrix}, \quad \Rightarrow \quad \begin{matrix} r_i, \mathrm{g}_i, \mathrm{b}_i = r_i, \mathrm{g}_i, \mathrm{b}_i - \delta, \\ p_i = r_i, \mathrm{g}_i, \mathrm{b}_i, \end{matrix} \tag{3}$$

где p_i – пиксел картинки, $r_i, \mathrm{g}_i, \mathrm{b}_i$ – компонента цвета пикселя p_i $(r_i, \mathrm{g}_i, \mathrm{b}_i = [0, 255])$, Y_i – яркость пикселя, g_{max} – максимальное возможное значение яркости, $g_{max} = 255, g_{min}$ – минимально возможное значение яркости, $g_{min} = 0$, $\Delta_g = g_{max} - g_{min}$, gg_i – новая яркость пикселя, где $i = \overline{1,n}$, n – количество пикселей в изображении.

Ниже представлен код реализации линейного контрастирования:

…

```
1.  rgb:=ColorToRgb(BitMap.Canvas.Pixels[x,y]);
2.  r := getRvalue(rgb);
3.  g := getGvalue(rgb);
4.  b := getBvalue(rgb);
5.  f:=round((0.30*r)+(0.59*g)+(0.11*b));
6.  gg:=round((((f-fmin)/(fmax-fmin))*(gmax-gmin))+gmin);
7.  raz:=f-gg;
8.  r :=BLimit(r-raz);
9.  g :=BLimit(g-raz);
10. b :=BLimit(b-raz);
11. BitMap.Canvas.Pixels[x,y]:=windows.RGB(r,g, b);
```

1.9. Шум

Цифровой шум проявляется на изображении в виде наложения маски из пикселей случайного цвета или яркости.

Шум накладывается для того, чтобы проверить работу фильтров шумоподавления. Шум, который реализован в данной работе, называется «соль-перец» (Рисунок 1.7.), в связи с тем, что случайным образом на изображение накладываются точки черного и белого цвета.

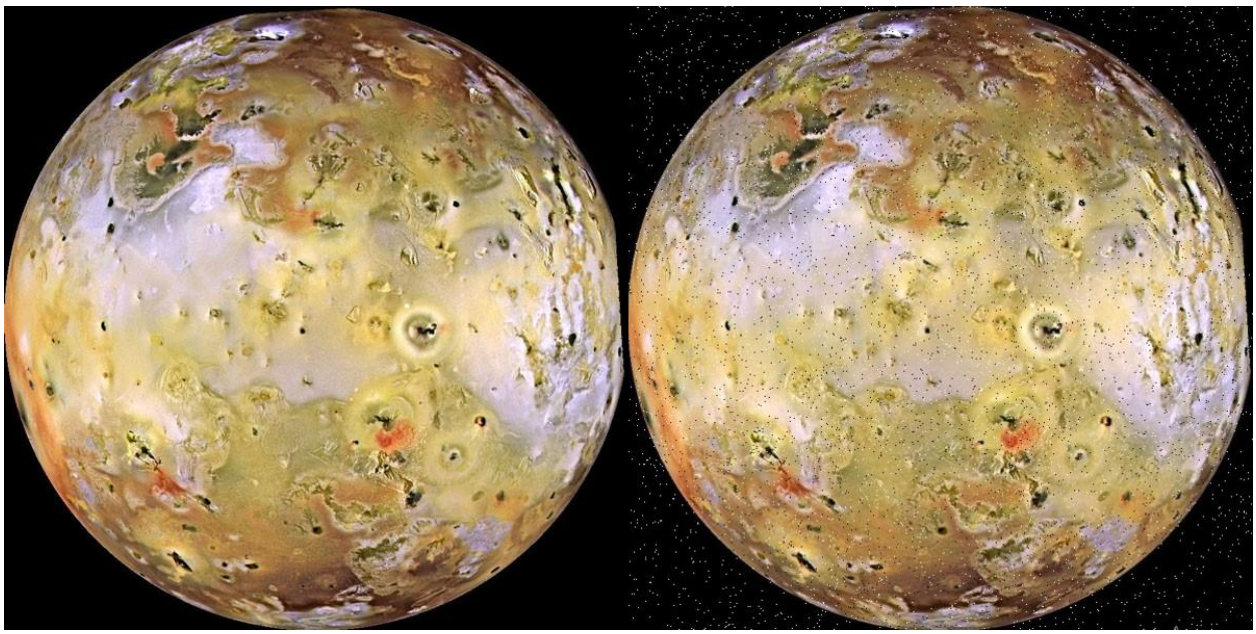

Рисунок 1.7. – Шум

Ниже представлен код реализации искусственного зашумления:

```
for i := 1 to 1000 do begin
x := Random (BitMap.Width);
y := Random (BitMap.Height);
BitMap.Canvas.Pixels [x, y] := ClBlack; end;
for i := 1 to 1000 do begin
x := Random (BitMap.Width);
y := Random (BitMap.Height);
BitMap.Canvas.Pixels [x, y] := ClWhite; end;
```

1.10. Сглаживающий фильтр

Апертура фильтра – это размер окна (части изображения), с которым фильтр работает непосредственно в данный момент времени; окно это постепенно передвигается по изображению слева направо и сверху вниз на один пиксель (то есть на следующем шаге фильтр работает с окном, состоящим не только из элементов исходного изображения, но и из элементов, ранее подвергнувшихся преобразованию, – своего рода «принцип снежного кома»).

Сглаживающие фильтры действуют на изображение аналогично мутному стеклу: изображение становится нерезким, размытым.

Сглаживающий фильтр основывается на следующем принципе: находится среднее арифметическое значение всех элементов рабочего окна изображения (отдельно по каждому из каналов), после чего это среднее значение становится значением среднего элементом (для двумерного случая средним элементом будет средний элемент по горизонтали и вертикали, то есть центр квадрата).

Сглаживающий фильтр учитывает точку, в которой присутствует шум, поэтому для полного устранения шума, его необходимо применять несколько раз, при этом изображение все больше становится размытым (Рисунок 1.8., 1.9.).

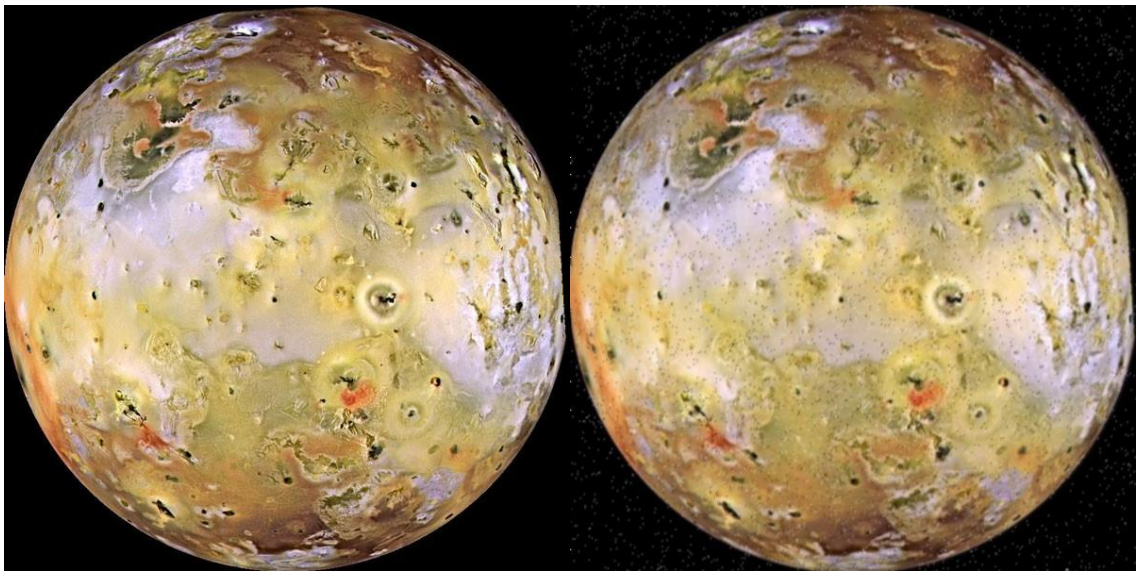

Рисунок 1.8. – Сглаживающий фильтр, примененный один раз

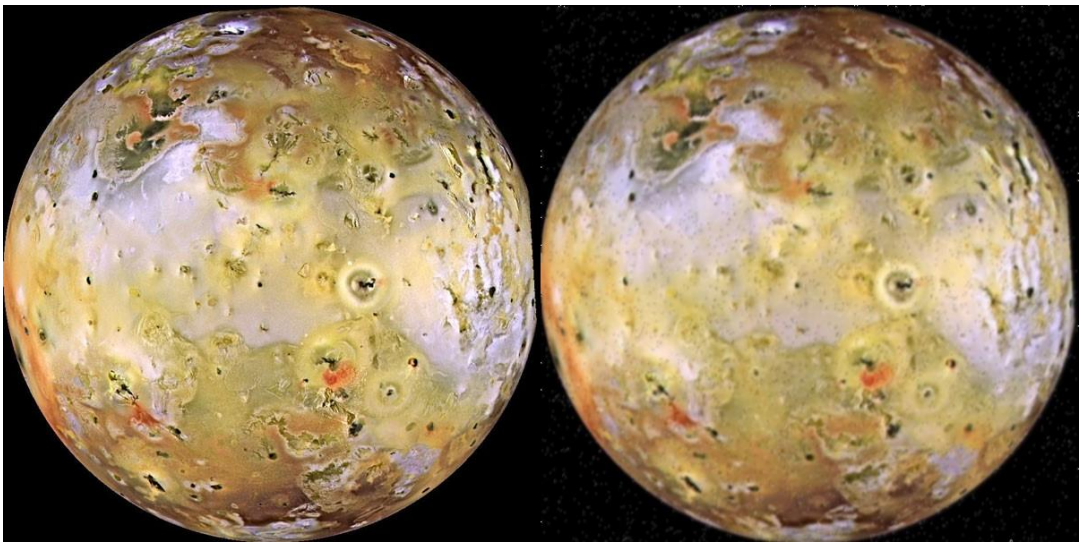

Рисунок 1.9. – Сглаживающий фильтр, примененный дважды

1.11. Медианный фильтр

Медианный фильтр (Рисунок 1.10.) основывается на нахождении медианы – среднего элемента (но не среднего арифметического) последовательности в результате её упорядочения по возрастанию/убыванию и присваиванию найденного значения только среднему элементу (речь снова о нечётной апертуре). Например, для той же апертуры 3 и двумерного фильтра мы должны упорядочить 9 точек (например, по возрастанию), после чего значение 5-ой точки упорядоченной последовательности отправить в центр окна фильтра (3x3). Для упорядочения можно использовать любой из известных методов сортировки, мы использование сортировку методом пузырька:

```
1. for j:=Left to Right-1 do
2. for i:=Left to Right-1 do
3. if Bytes[i]>Bytes[i+1] then begin
4. tmp:=Bytes[i];
5. Bytes[i]:=Bytes[i+1];
6. Bytes[i+1]:=tmp; end;
```

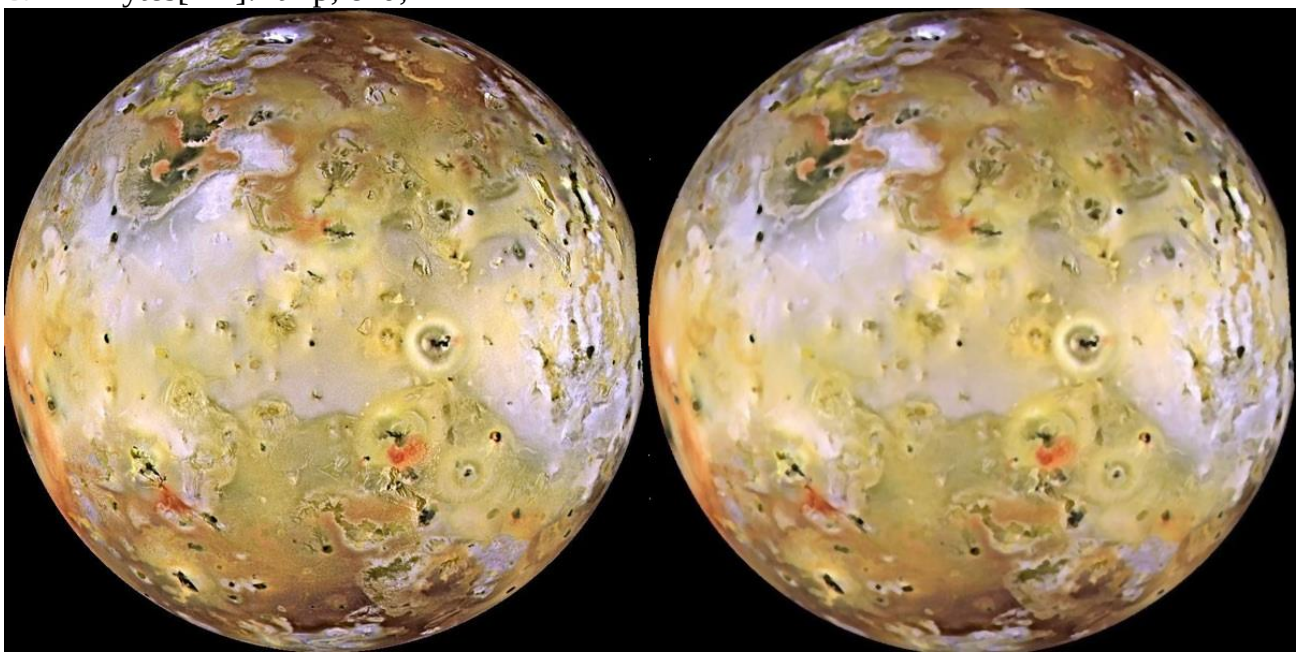

Рисунок 1.10. – Медианный фильтр

2. Разработка ПО Koenigsberg 1.1

2.1. Описание Koenigsberg 1.1

Для создания приложения, с помощью которого производится обработка цифровых изображений, использовалась среда Turbo Delphi 2006.

Данное программное обеспечение работает с форматами BMP и JPEG/JPG.

На форму, помещен компонент MainMenu, расположенный на вкладке Standart. Данный компонент не визуальный, т.е. в процессе проектирования место его размещения на форме не представляет никакого значения, в связи с тем, что мы увидим меню, сгенерированное нами, а не сам компонент. Основное свойство данного компонента – Items, которое заполняется с помощью Конструктора Меню, вызываемое двойным щелчком левой кнопкой мыши на MainMenu или нажатием кнопки многоточием рядом со свойством Items в окне Инспектора Объектов[1] [9].

Рисунок 2.1. – Инспектор Объектов

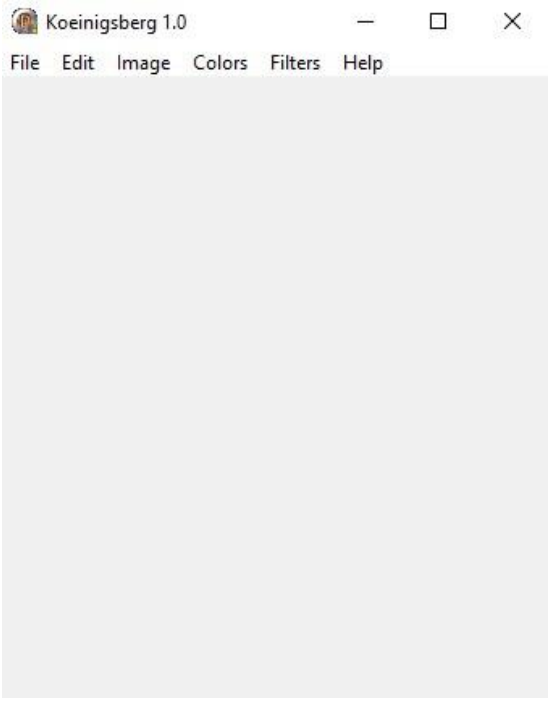

Рисунок 2.2. – Главное окно ПО

[1] Инспектор Объектов (Object Inspector) – основной инструмент (Рисунок 2.1.), с помощью которого вы будете задавать свойства компонентов и писать обработчики событий. Окно Инспектора Объектов имеет две страницы. В верхней части окна расположен выпадающий список всех компонентов, помещенных на форме. В нем вы можете выбрать тот компонент, свойства и события которого вас интересуют [10].

Компонент MainMenu заполнен следующими разделами (Рисунок 2.2.):

- File
- Edit
- Image
- Color
- Filters
- Help.

2.2. Меню ПО Koenigsberg 1.1

2.2.1. File

File включает в себя разделы (Рисунок 2.3.):

- Open
- Save
- Save As
- Close
- Exit.

Рисунок 2.3. – Раздел File

Основным событием каждого элемента свойства Items является OnClick[2].

[2] Событие OnClick возникает при нажатии левой кнопкой мыши.

Для написания обработчика события OnClick на элемент Open помещен на форму компонент OpenPictureDialog[3], расположенный на вкладке Dialogs. После выполнения метода Execute, получаем полный адрес выбранного файла. С помощью функций ExtractFileExt получен формат файла, и в зависимости от него производится преобразование его в единый формат, с которым работает данная программа. В качестве единого формата был взят bmp-формат. Преобразованный файл изображения загружен в компонент Image, который в программе назван ImageMain. Image обеспечивает вывод на поверхность формы иллюстраций, представленных в *bmp*-формате. Он расположен на вкладке Additional [9]. У ImagemMain в свойстве AutoSize стоит true, где размер данного компонента автоматически подгоняется под размер загруженного изображения. В свойствах Top и Width поставлено значение равное 0, это необходимо для реализации вычисления и вывода каждого пикселя на экран, т.е. компонент не будет иметь ширину и высоту, пока он пуст.

Аналогичным образом происходит сохранение (Save As) изображения на жёсткий диск, только вместо OpenPictureDialog используется SavePictureDialog[4], расположенный на той же вкладке Dialogs и помешенный на форму. При сохранении в файл идет обратное преобразование единого формата в необходимый формат изображения.

Save сохраняет файл используя данные последнего вызова одного из этих диалогов. В свойстве ShortCut, установлена комбинация горячих клавиш CTRL + S, которая соответствует стандартному сохранению.

Close отчищает компонент ImageMain. У данного компонента есть свойство Picture, которое содержит изображение. Данному свойству присваивается значения nill, т.е. нулевое значение.

Для Exit, который отвечает за закрытие программы, в обработчике событий OnClick написано close.

[3] Компонент OpenPictureDialog предназначен для открытия графических файлов.
[4] Компонент SavePictureDialog предназначен для сохранения графических файлов.

2.2.2. Edit

Edit включает в себя разделы (Рисунок 2.4.):

- Undo
- Redo.

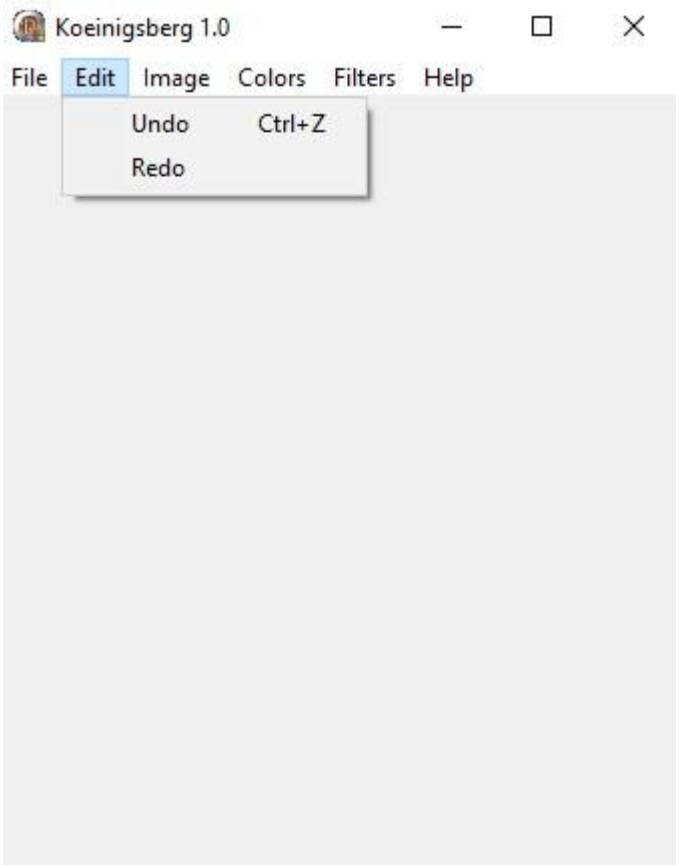

Рисунок 2.4. – Раздел Edit

Для реализации Undo и Redo создан класс[5] динамического массива типа TBitmap, включающий свойства: Place (место, где картинка должна отрисоваться, т.е. ImageMain) типа TImage, ActiveNum (активный номер, т.е. номер изображения, который выведен на экран) типа integer, ImageArray (массив, в который добавляются и удаляются изображения) типа array of

[5] Класс — это тип данных, который определяется пользователем и имеет собственные поля, методы и свойства. Фактически, класс является дальнейшим развитием уже знакомого нам типа данных - записи. Но если запись имела только свойства (поля) и ничего более, то классы могут содержать все, что требуется для полноценного объекта. При этом класс может быть сколь угодно большим и иметь в качестве своих свойств другие классы. Основными концепциями, применимыми к классам являются инкапсуляция, наследование и полиморфизм. Так, инкапсуляция, или объединение в единое целое всего того, что относится к определенному объекту, на практике выливается в возможность объединить в одном объекте свойства и методы. Наследование позволяет, единожды создав некий обобщенный класс, пользоваться его свойствами и методами из других классов, созданных на его основе - в классах-потомках. Ну а применение полиморфизма на практике позволит переопределить свойства или изменить поведение отдельных методов родительского класса в этих самых классах-потомках [11].

MyTBMP, MyTBMP тип идентичный классу TBitmap), а также методы: constructor Create(Inplace:TImage), т.е. конструктор; procedure AddElem(Image:MyTBMP) – создание нового элемента и копирование изображения, которое передается, в массив изображений; procedure Undo() – назад (отменить действие); procedure Redo() – вперед (вернуть действие); procedure Rendering() – отрисовка изображения; function GetBMP:MyTBMP – получить изображение.

2.2.3. Image

Image включает в себя разделы (Рисунок 2.5.):

- Image information
- Histograms.

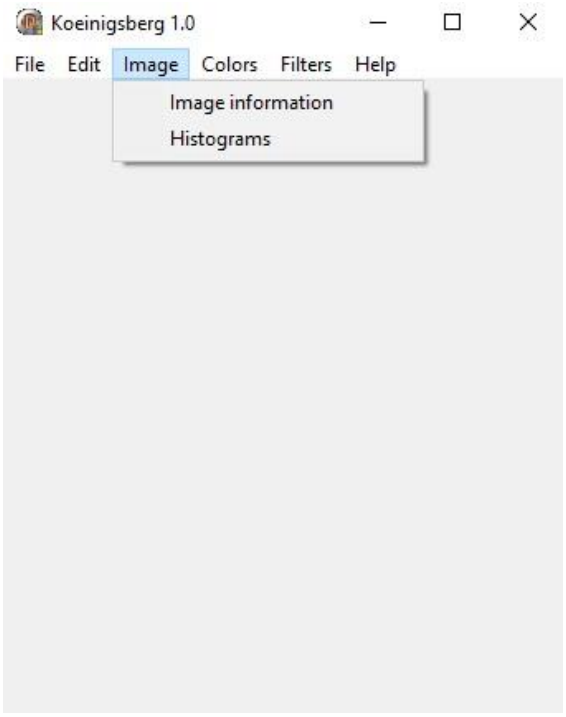

Рисунок 2.5. –Раздел Image

Раздел Image information содержит информацию о загруженном изображении (Рисунок 2.6.).

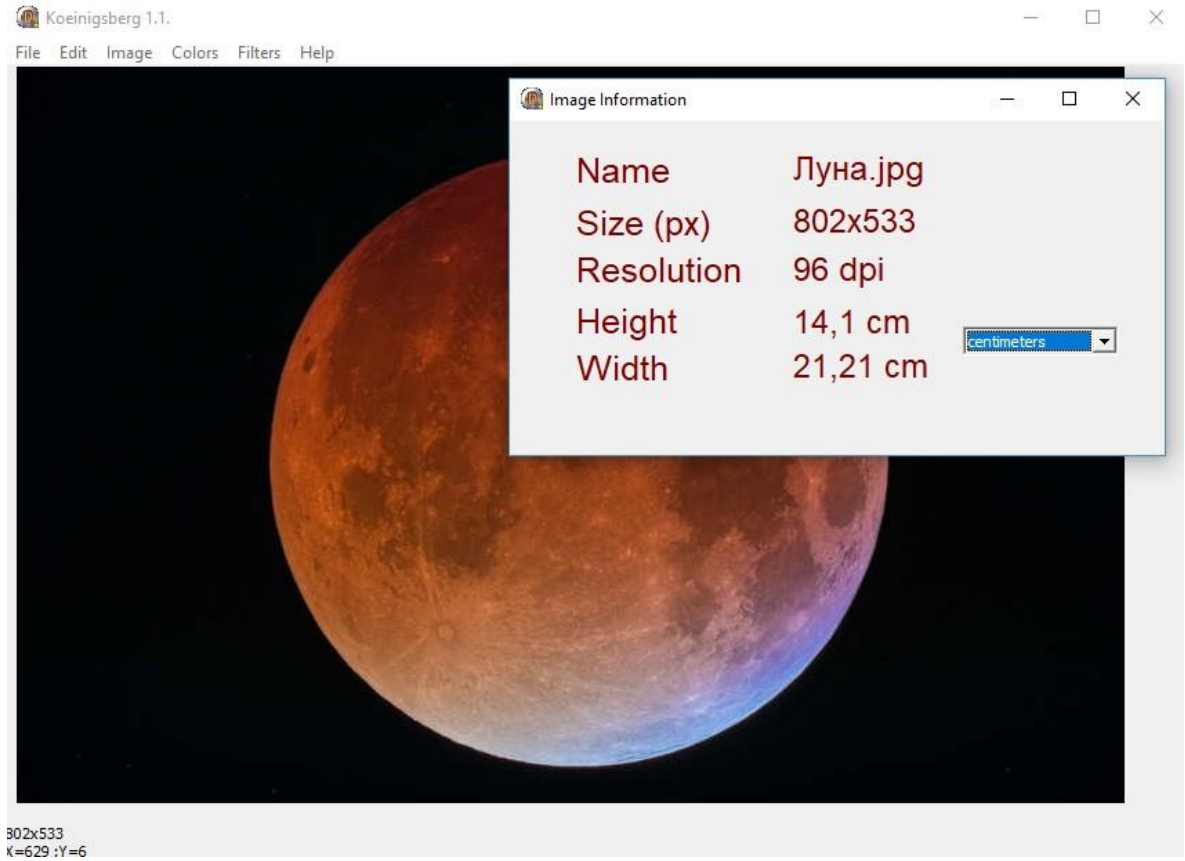

Рисунок 2.6. – Окно Image Information

Создается новая форма, на которую помещаются десять компонентов Label и один ComboBox. У пяти компонентов Label в свойство Caption заносится информация, которая будет отображаться на форме, т.е. Name, Size, Resolution, Height, Width, остальные пять несут, соответственно, информационные значения (Рисунок 2.6.). В свойстве Font устанавливаем параметры выводимого текста, т.е. цвет, размер и стиль шрифта (цвет – clMaroon, размер – 19, стиль шрифта – Arial). ComboBox нужен для вывода единиц измерения высоты и ширины изображения, т.е. сантиметры или дюймы. Отличительным свойством является Style, который при значении csDropDownList запрещает вводить информацию в строке ввода, а может принимать значения только одной из строк, сохранённых в компоненте. В свойство Items заносятся две строки: centimeters и inches, по умолчанию стоят сантиметры, которым соответствует индекс 0, заносящийся в свойство ItemIndex. В обработчике событий OnShow (он происходит при отображении формы) для получения имени файла без имени диска и каталогов

используется функция ExtractFileName, в параметрах которой указывается полное имя файла, в нашем случае FormMain.Filename. Размер изображения в пикселях вычисляется следующим образом: с помощью функции преобразования числа в строку (IntToStr), узнаем ширину и высоту загруженного изображения, и поставим между ними ‘x’ после нахождения. Разрешение помогает узнать функция PixelsPerInch (количество точек на дюйм), но, чтобы вывести числовое значение используется функция IntToStr. При нахождении высоты и ширины изображения учитывается то, в каких единицах нам необходима информация. В обработчике событий OnChange компонента ComboBox используется оператор выбора case. Ниже представлен код:

```
1.  case INCHSM.ItemIndex of
2.  1: begin
3.  HeightInformation.Caption:=FloatToStr(Trunc((FormMain.ImageMain.Picture.
    Height/FormMain.PixelsPerInch)*100)/100)+' inch';
4.  WidthInformation.Caption:=FloatToStr(Trunc((FormMain.ImageMain.Picture.
    Width/FormMain.PixelsPerInch)*100)/100)+' inch';
5.  end;
6.  0: begin
7.  HeightInformation.Caption:=FloatToStr(Trunc((FormMain.ImageMain.Picture.
    Height/FormMain.PixelsPerInch)*100*2.54)/100)+' cm';
8.  WidthInformation.Caption:=FloatToStr(Trunc((FormMain.ImageMain.Picture.
    Width/FormMain.PixelsPerInch)*100*2.54)/100)+' cm';
9.  end;
10. end;
```

ItemIndex=1 соответствует дюймам: FormMain.ImageMain.Picture.Height – высота изображения, загруженного в компонент ImageMain, FormMain.PixelsPerInch – число точек на дюйм, Trunc – функция, предназначенная для отбрасывания дробной, FloatToStr – функция преобразовывает число с плавающей запятой в его строковое представление, FormMain.ImageMain.Picture.Width – ширина изображения, загруженного в компонент ImageMain. ItemIndex=0 соответствует сантиметрам: отличие от

дюймов заключается в том, что мы умножаем выражение на 2.54 (т.к. связь сантиметров и дюймов определяется соотношением 1 дюйм = 2.54 см). Метод ComboBox.OnCange подается в событие OnShow формы. Данная форма вызывается как немодальное окно с помощью метода Show, в обработчике событий OnClick элемента ImageInformation.

Раздел Histograms содержит гистограммы изображения по трем каналам (Рисунок 2.7., Рисунок 2.8., Рисунок 2.9.).

Создается новая форма, на которую помещаются компоненты Chart, ComboBox и Label. У Label в свойстве Caption вводится Channel. В свойстве Font устанавливаем параметры выводимого текста, т.е. цвет, размер и стиль шрифта (цвет – clBlack, размер – 13, стиль шрифта – Arial). При двойном нажатии правой кнопкой мыши на компонент Chart отрывается окно Editing His, в котором указываются серии, которые будут использоваться в данном компоненте. Мы с помощью кнопки Add добавляем из вкладки Standart серию Area, предварительно убрав галочку рядом с 3D. Во вкладках Legend, Titles, Axis убираем галочки рядом с Visible, чтобы не отображать легенду, название осей. ComboBox необходим для того, чтобы иметь выбор просмотра гистограммы каждого канала. В свойство Items заносятся три строки: Red, Green, Blue. В свойстве ItemIndex устанавливаем индекс -1 для того, чтобы по умолчанию не строилась ни одна из вышеперечисленных гистограмм. В обработчике событий OnChange компонента ComboBox используется оператор выбора case. Ниже представлен код:

```
1.  case Channel.ItemIndex of
2.  0: HistogramRed(Sender);
3.  1: HistogramGreen(Sender);
4.  2: HistogramBlue(Sender);
5.  end;
```

ItemIndex = 0 соответствует построению гистограммы красного канала, ItemIndex = 1 – зеленого канала, ItemIndex = 2 – синего канала.

В обработчике событий FormShow данной формы проверяется загружена ли картинка в главную форму. Если изображение не загружено, то

гистограммы не строятся, иначе строится гистограмма красного канала и затем вызывается метод ComboBox.OnCange для выбора следующего канала. Это действие можно реализовать следующим образом:

```
1.  if FormMain.ImageMain.Picture.Graphic = nil then
2.  His.Visible:=false
3.  else
4.  HistogramRed(Sender);
5.  ChannelChange(Sender);
6.  end;
```

Данная форма вызывается как немодальное окно с помощью метода Show, в обработчике событий OnClick элемента ImageInformation.

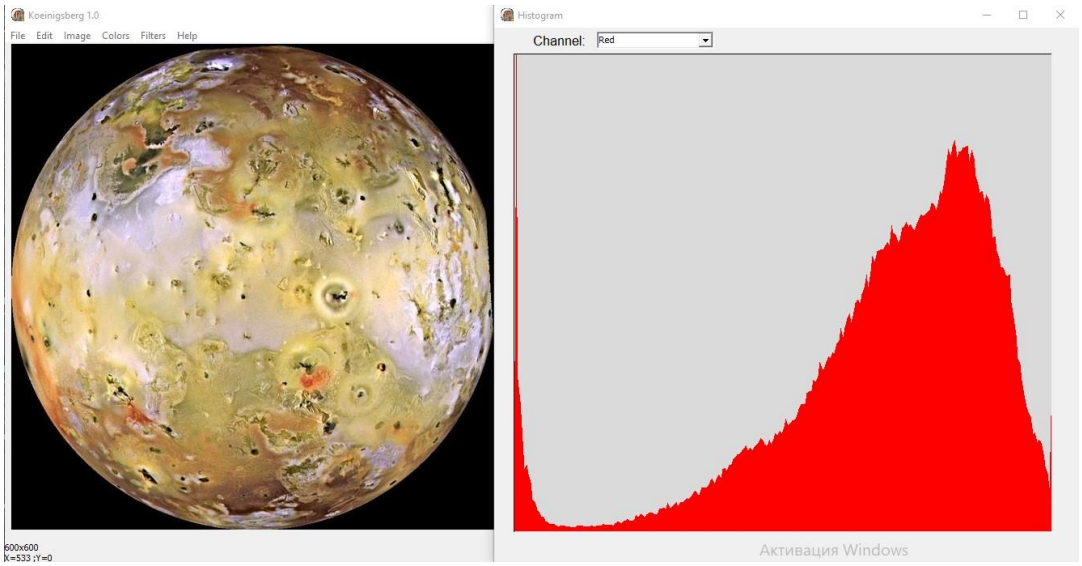

Рисунок 2.7. – Построение гистограммы красного канала

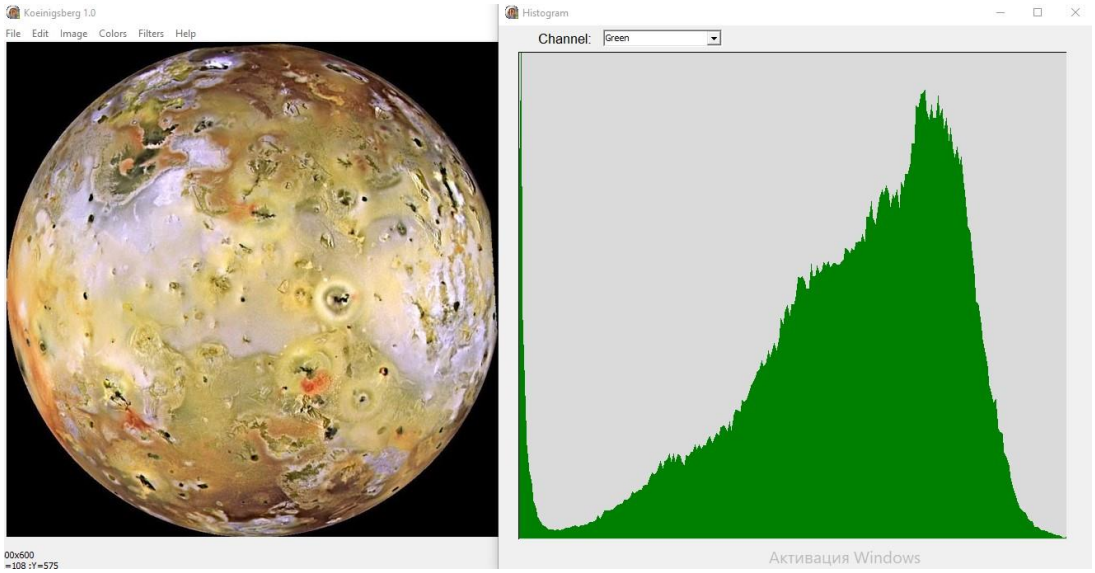

Рисунок 2.8. – Построение гистограммы зеленого канала

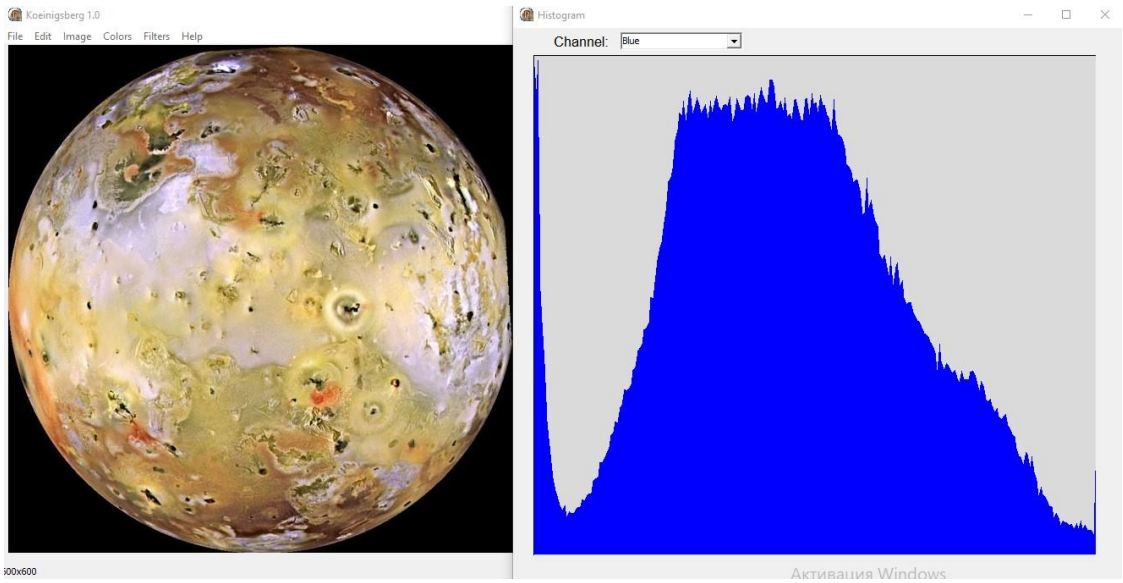

Рисунок 2.9. – Построение гистограммы синего канала

2.2.4. Colors

Colors включает в себя разделы (Рисунок 2.10.):

- Brightness/Contrast
- Chromaticity
  - Negative
  - Black and white
  - GrayScale
- Linear Contrast

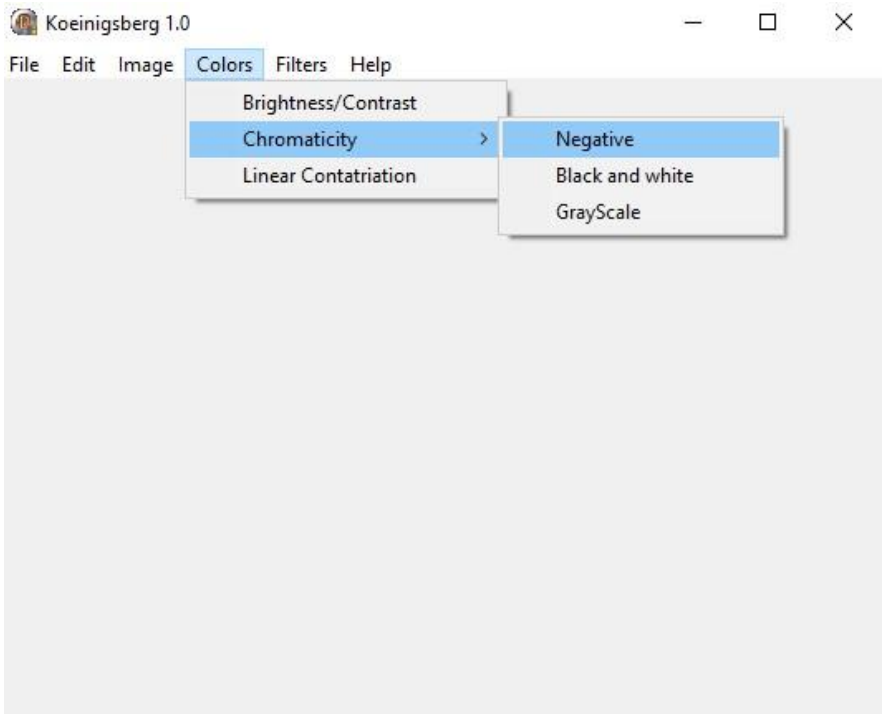

Рисунок 2.10. – Вкладка Colors

Для реализации Brigness/Contast создана форма, которая передается как модальное окно в обработчике событий OnClick главной формы. На форму помещены компоненты TrackBar для яркости и контрастности, а также три кнопки OK, Cancel и Apply. Компонент TrackBar представляет собой элемент управления в виде ползунка, который пользователь может перемещать курсором мыши. Основное свойство компонента – Position. При перемещении пользователем ползунка можно прочитать значение Position, характеризующее позицию, в которую пользователь переместил ползунок. Свойство Position – целое, значение которого может изменяться в пределах, задаваемых свойствами Min и Max [9]. По умолчанию, позиция ползунка стоит равная 0. Минимальное значение, которого может достигнуть ползунок принимается равное -100, максимальное – 100. Свойство Orientation определяет ориентацию ползунка [9]. Установлена горизонтальная ориентация, т.е. trHorizontal. Свойство TickMarcs указывает размещение шкалы относительно компонента [9]. Выбрано tmBottomRight – размещение снизу. В обработчике событий OnClose, т.е. при закрытии формы, установлены позиции ползунков равные 0.

В обработчике события OnClick кнопки Apply записан следующий код:

```
1. Ar.Undo();                  //отмена действия
2. Ar.AddElem(Ar.GetBMP); //добавление элемента в массив
3. EffectUnit.Brightness(Ar.GetBMP,TrackBrightness.Position);//изменение
   //яркости
4. EffectUnit.Contrast(Ar.GetBMP,TrackContrast.Position,False);//изменение
   //контрастности
5. Ar.Rendering();             //отрисовка изображения
```

Для кнопки Cancel:

```
1. Ar.Undo();          //отмена действия
2. Ar.Rendering();     //отрисовка изображения
3. close;              //закрытие окна
```

Для кнопки OK:

```
1. ApplyBrightness_Contrast.OnClick(Sender);   //вызывается метод Apply
2. close;              //закрытие окна
```

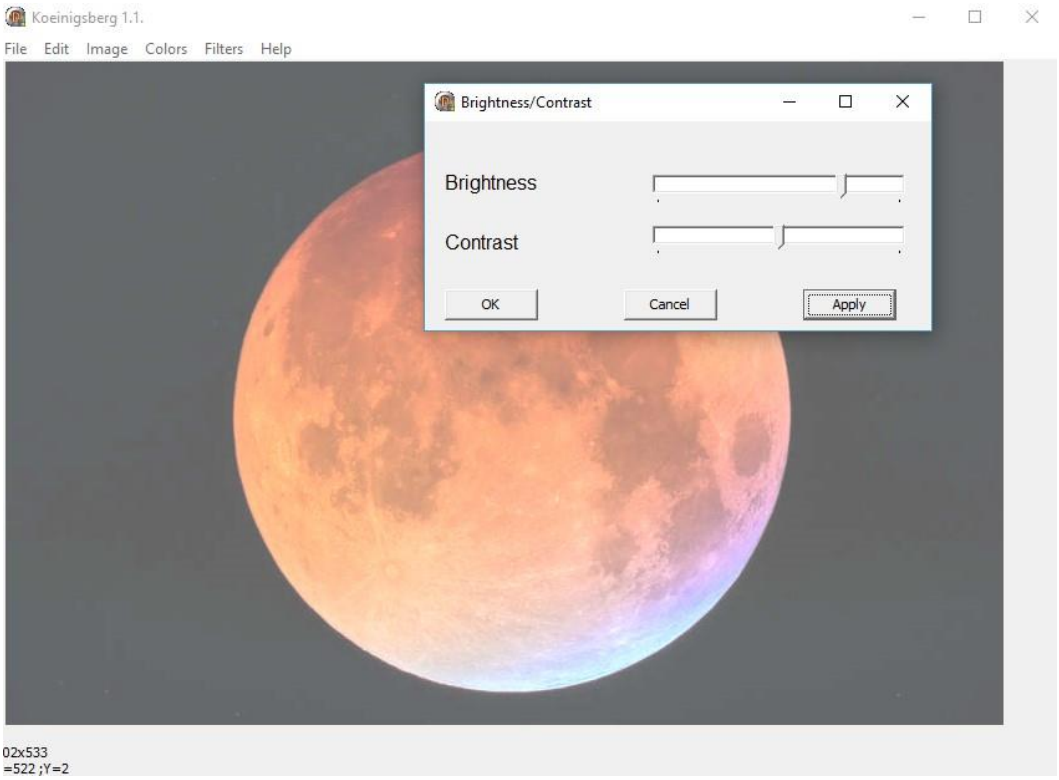

Рисунок 2.11. – Увеличение яркости в Koenigsberg 1.1

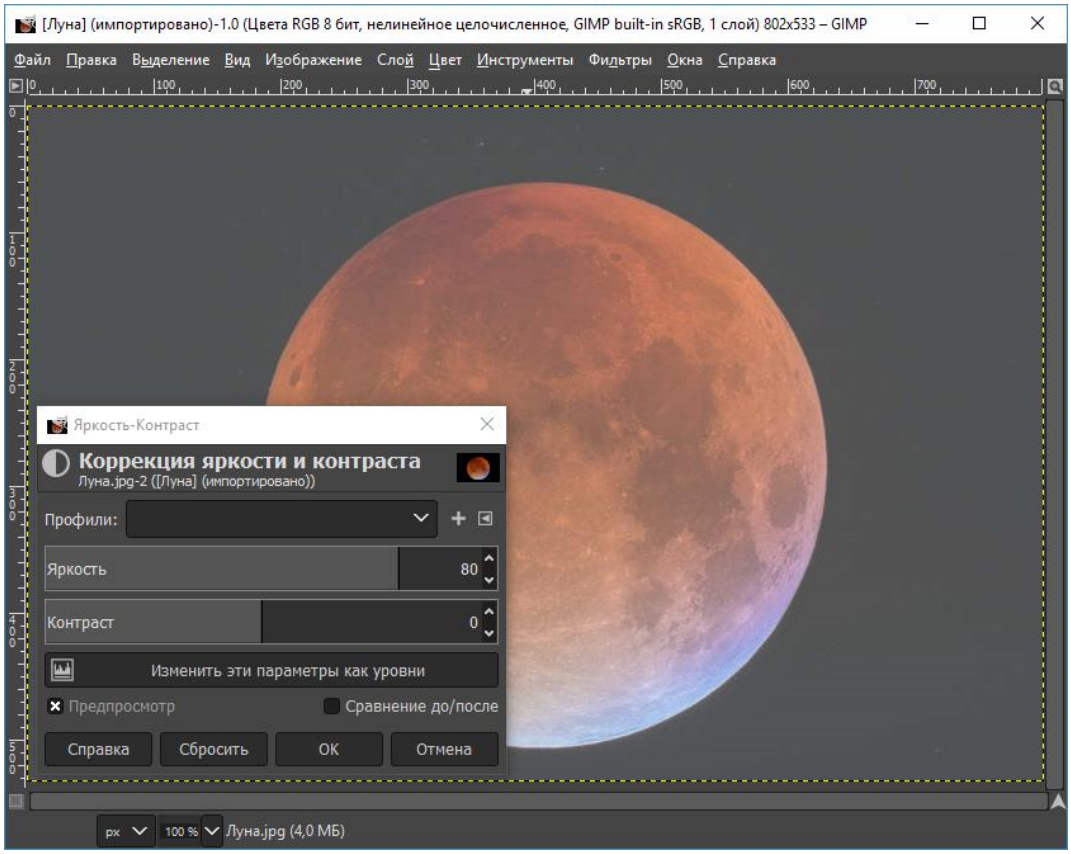

Рисунок 2.12. – Увеличение яркости в GIMP 2.10.10

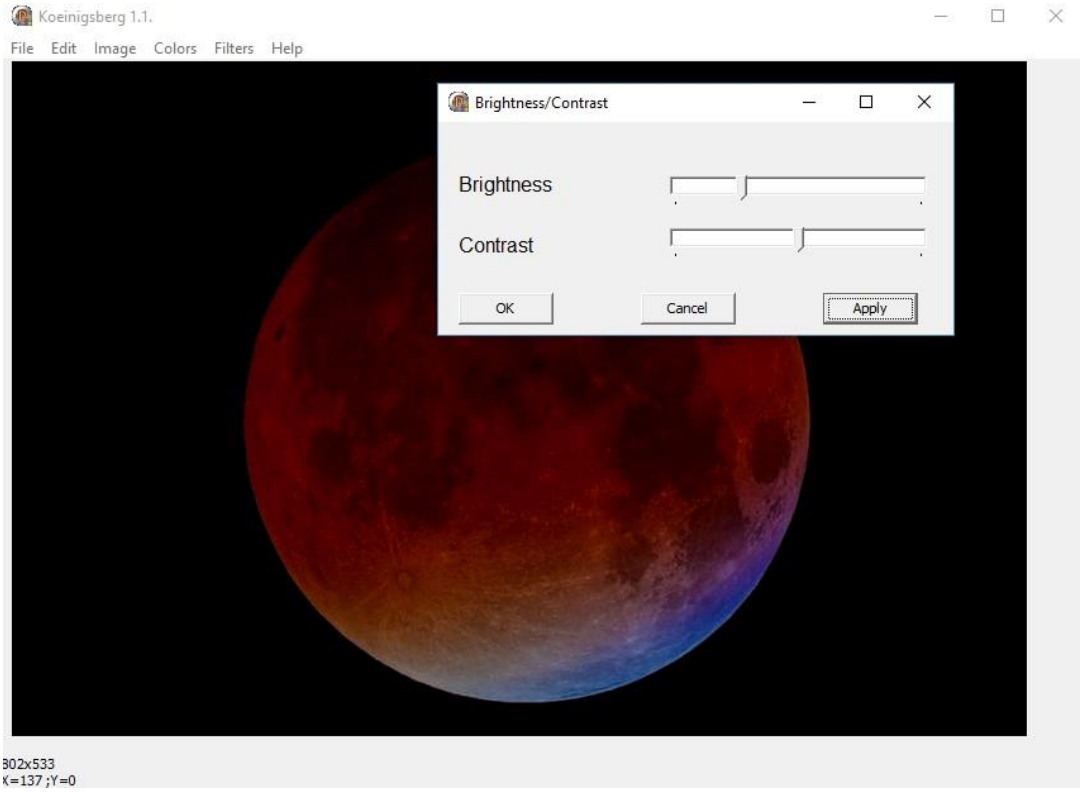

Рисунок 2.13. – Уменьшение яркости в Koenigsberg 1.1

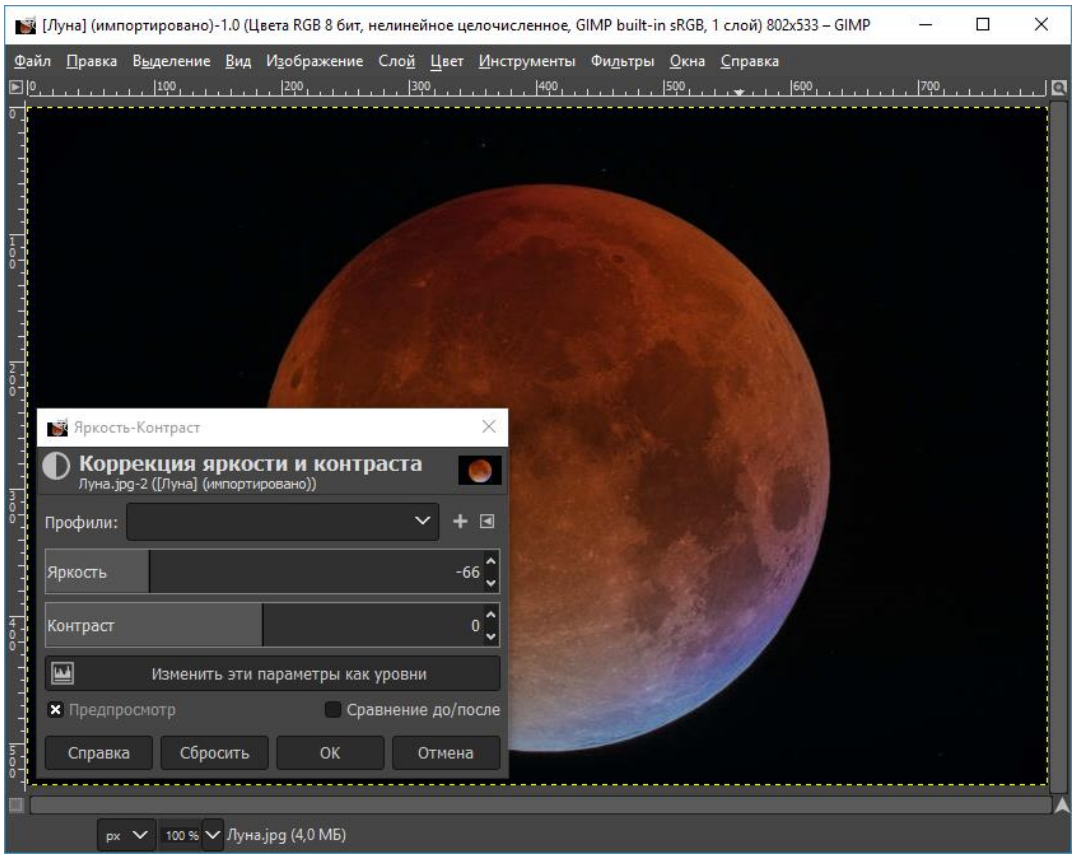

Рисунок 2.14. – Уменьшение яркости в GIMP 2.10.10

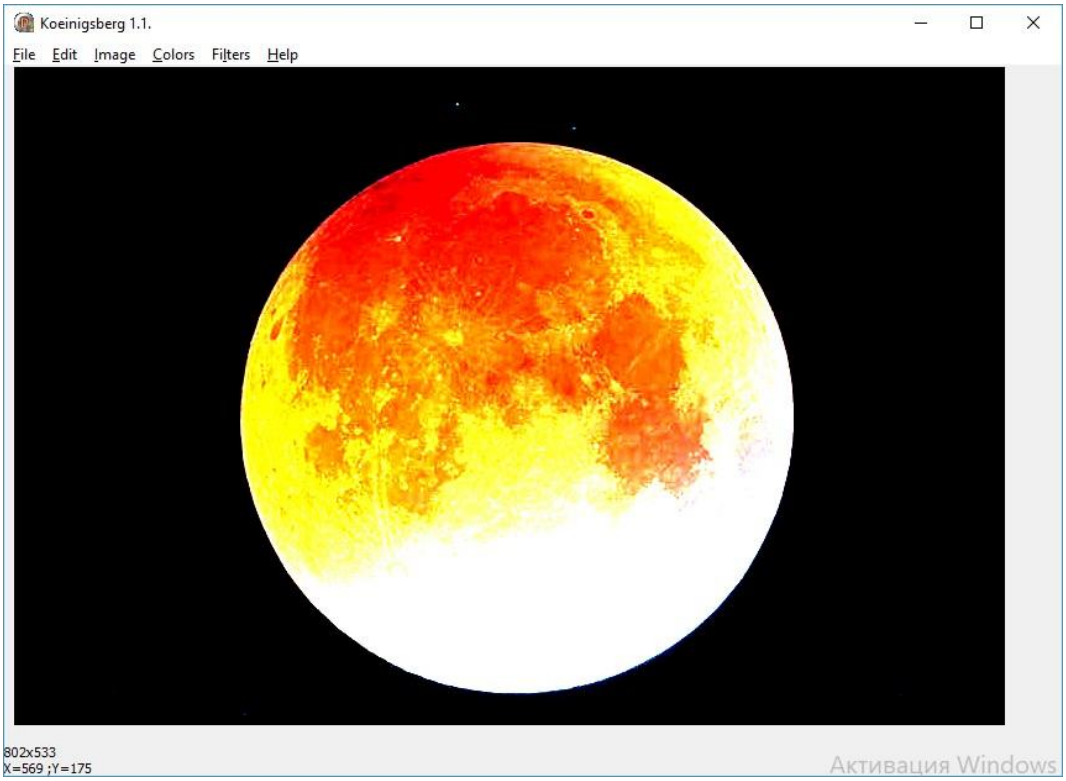

Рисунок 2.15. – Увеличение контрастности в Koenigsberg 1.1

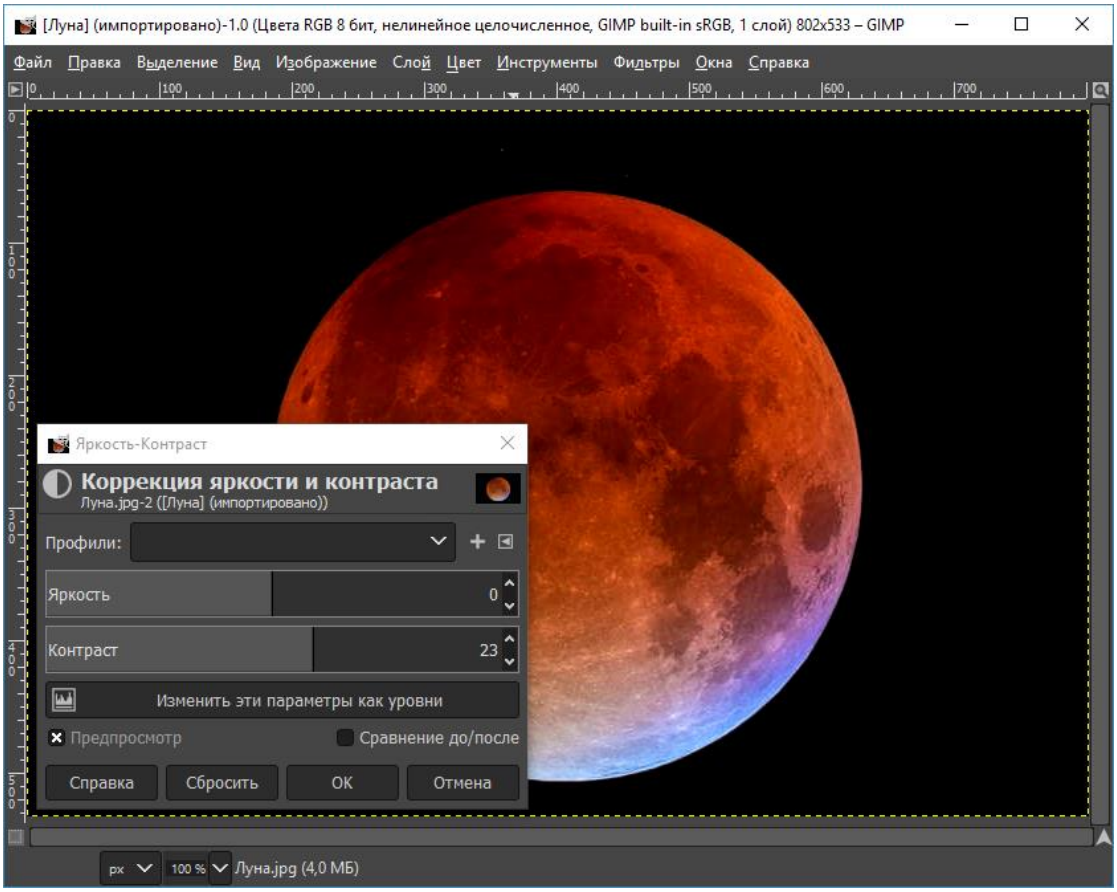

Рисунок 2.16. – Увеличение контрастности в GIMP 2.10.10

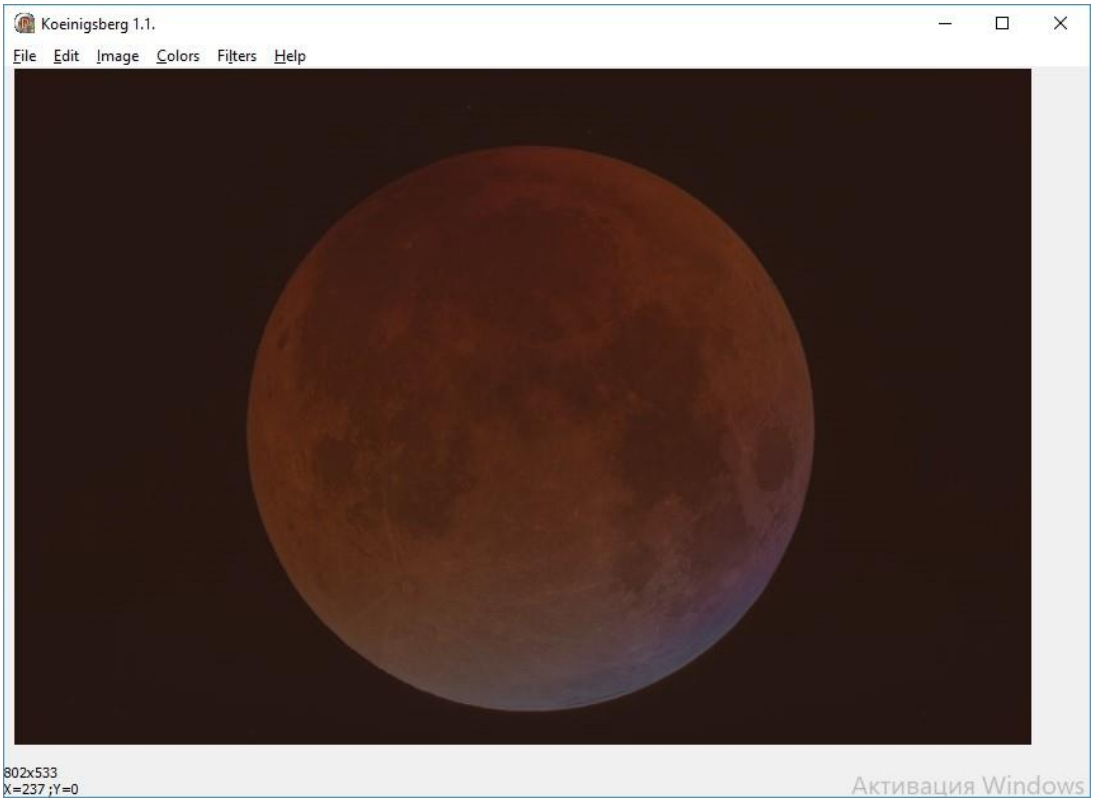

Рисунок 2.17. – Уменьшение контрастности в Koenigsberg 1.1

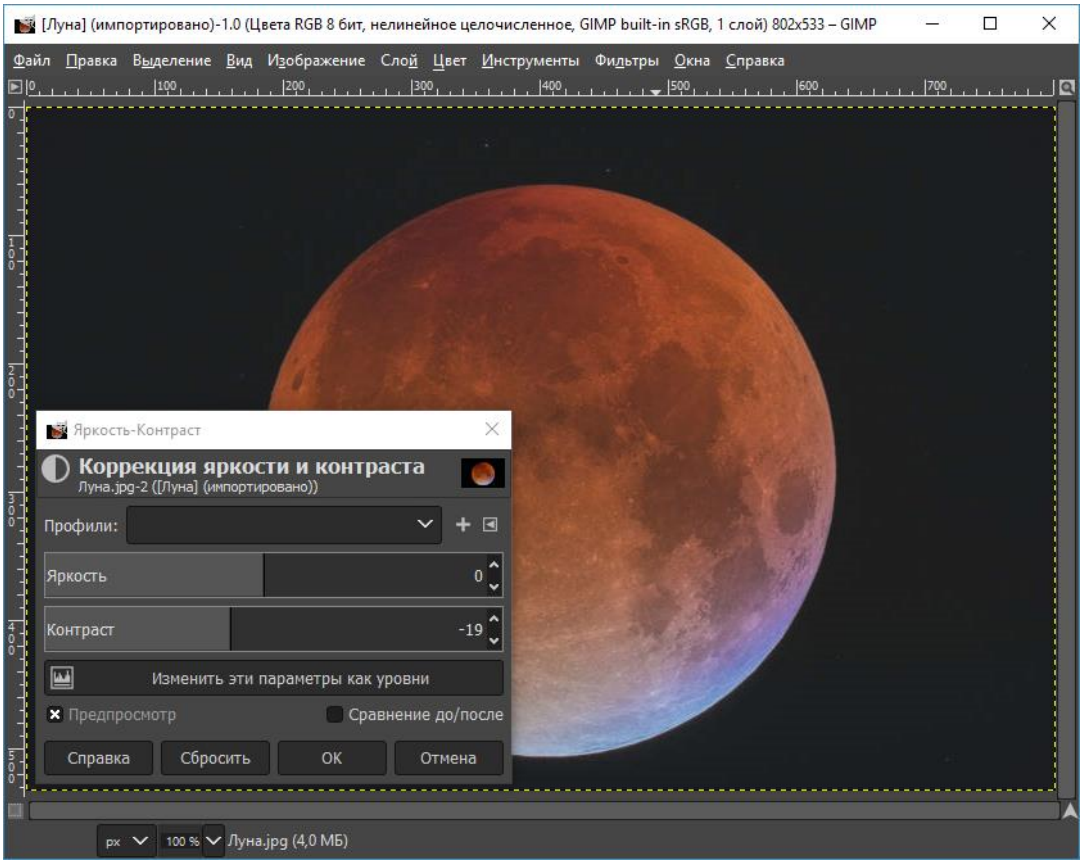

Рисунок 2.18. – Уменьшение контрастности в GIMP 2.10.10

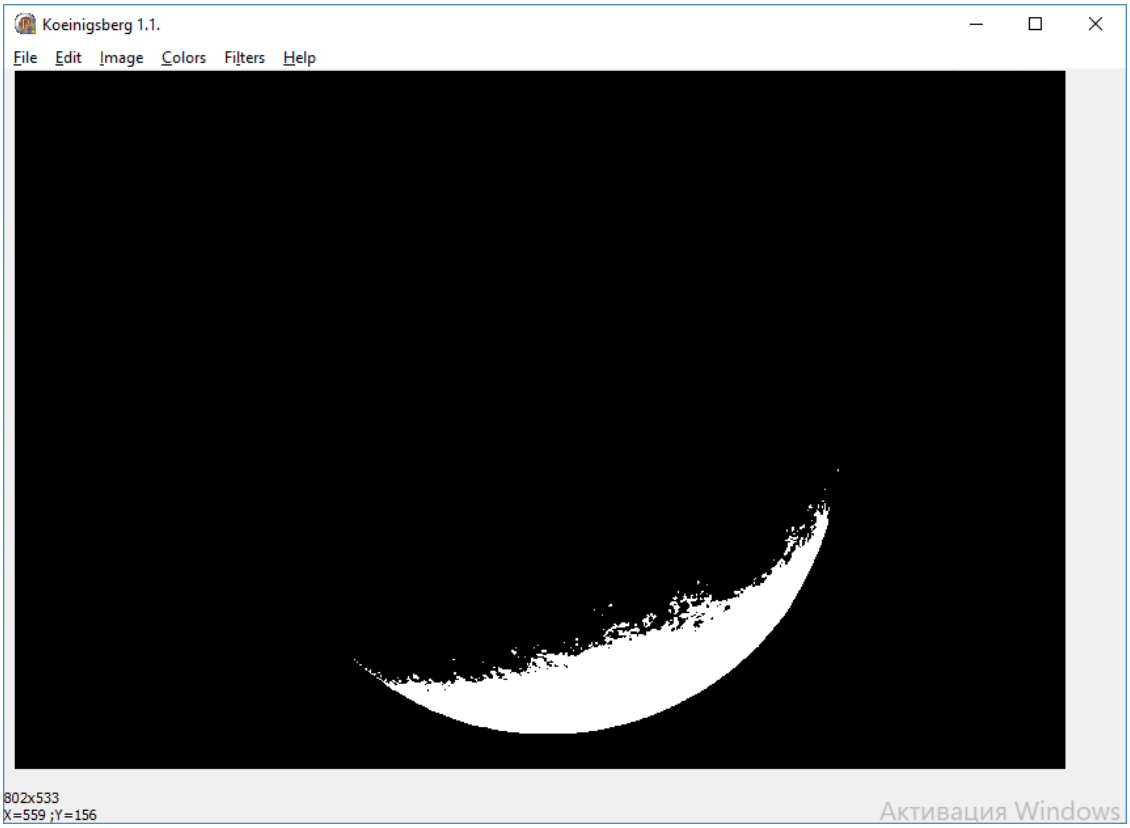

Рисунок 2.19. – Применение режима Black and White в Koenigsberg 1.1

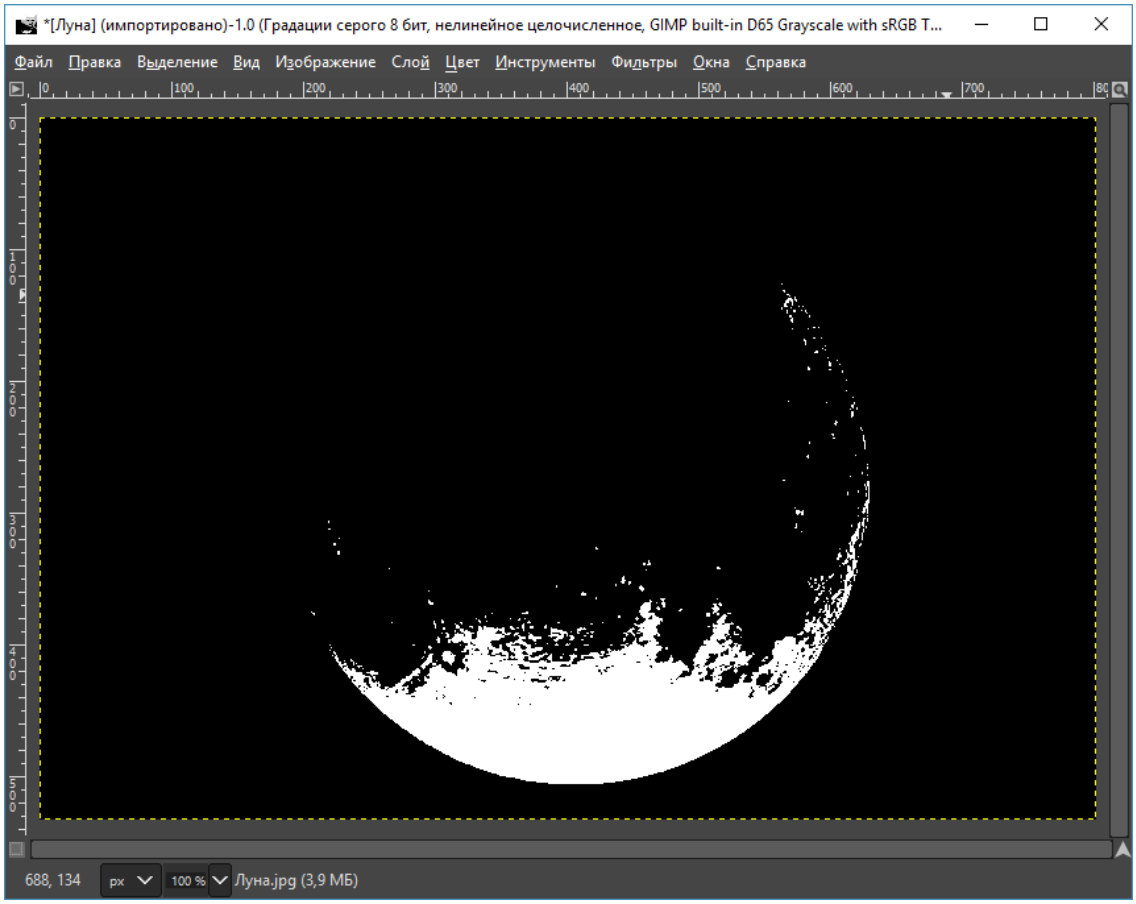

Рисунок 2.20. – Применение режима Black and White в GIMP 2.10.10

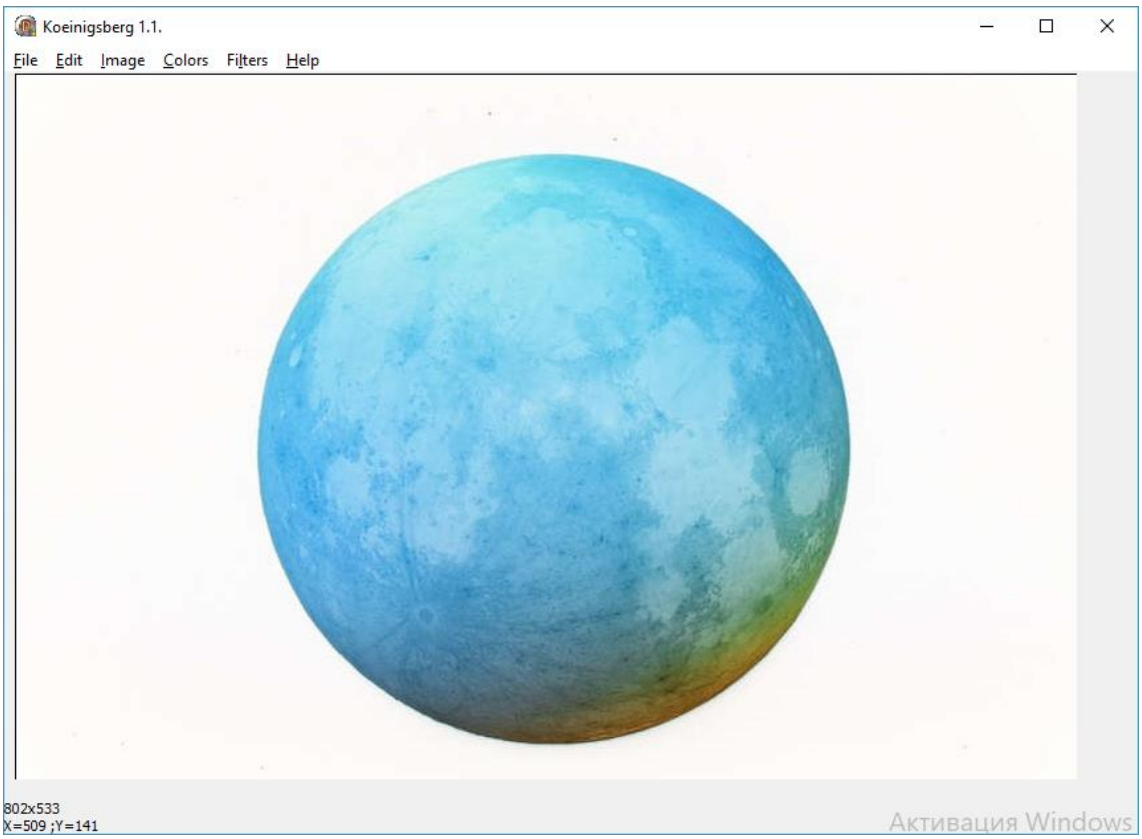

Рисунок 2.21. – Применение негатива в Koenigsberg 1.1

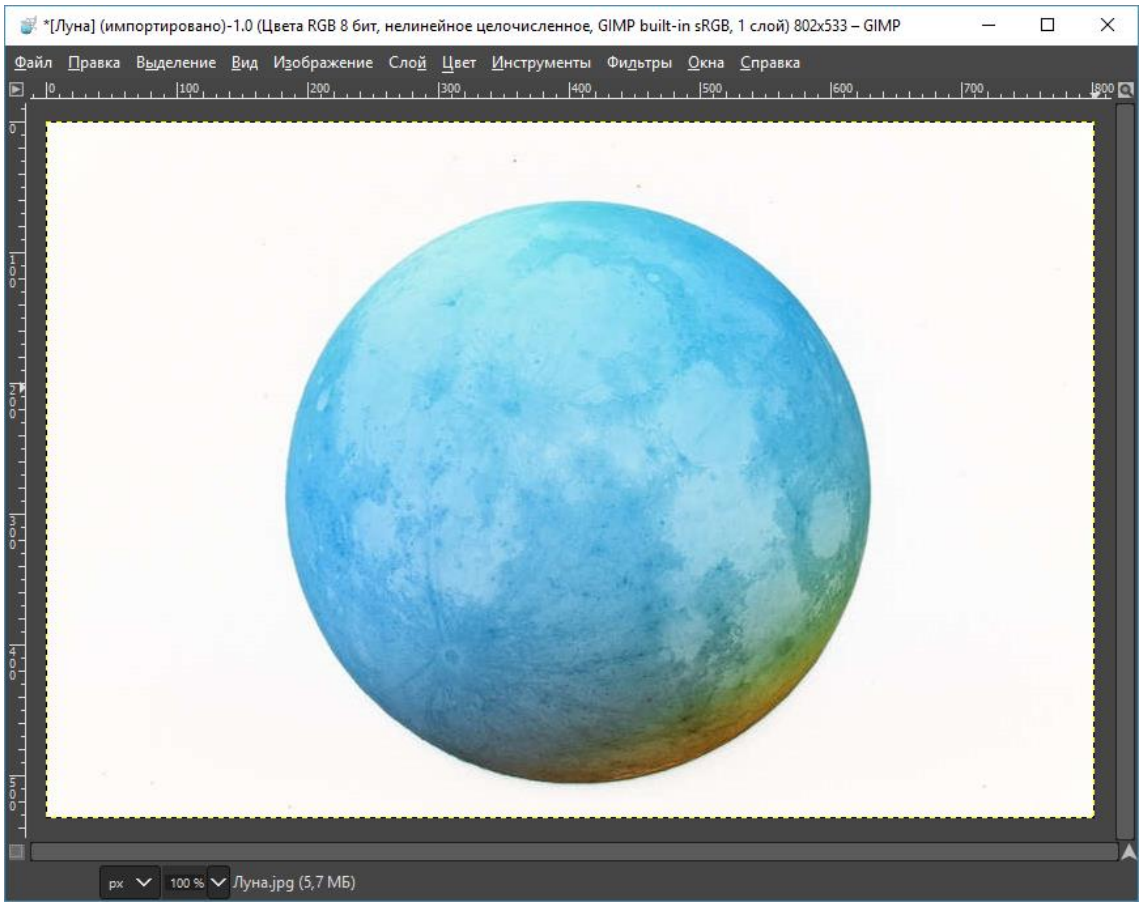

Рисунок 2.22. – Применение негатива в GIMP 2.10.10

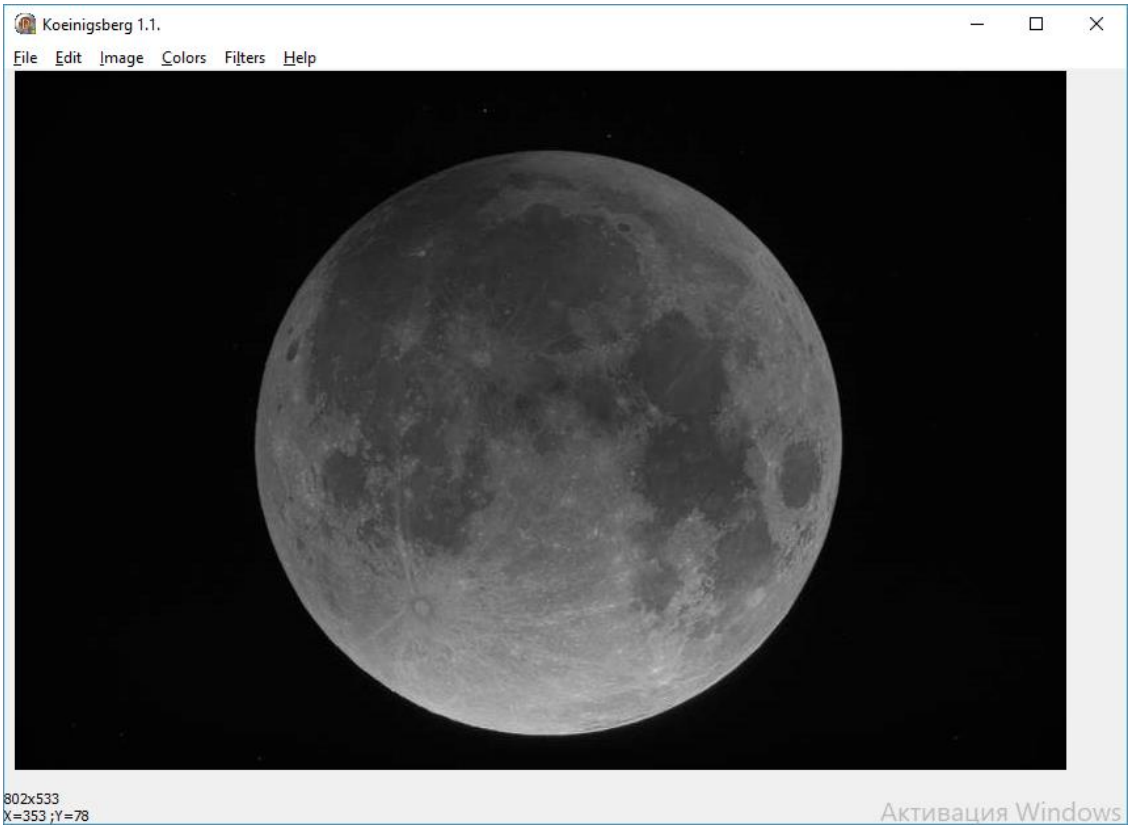

Рисунок 2.23. – Применение режима градации серого в Koenigsberg 1.1

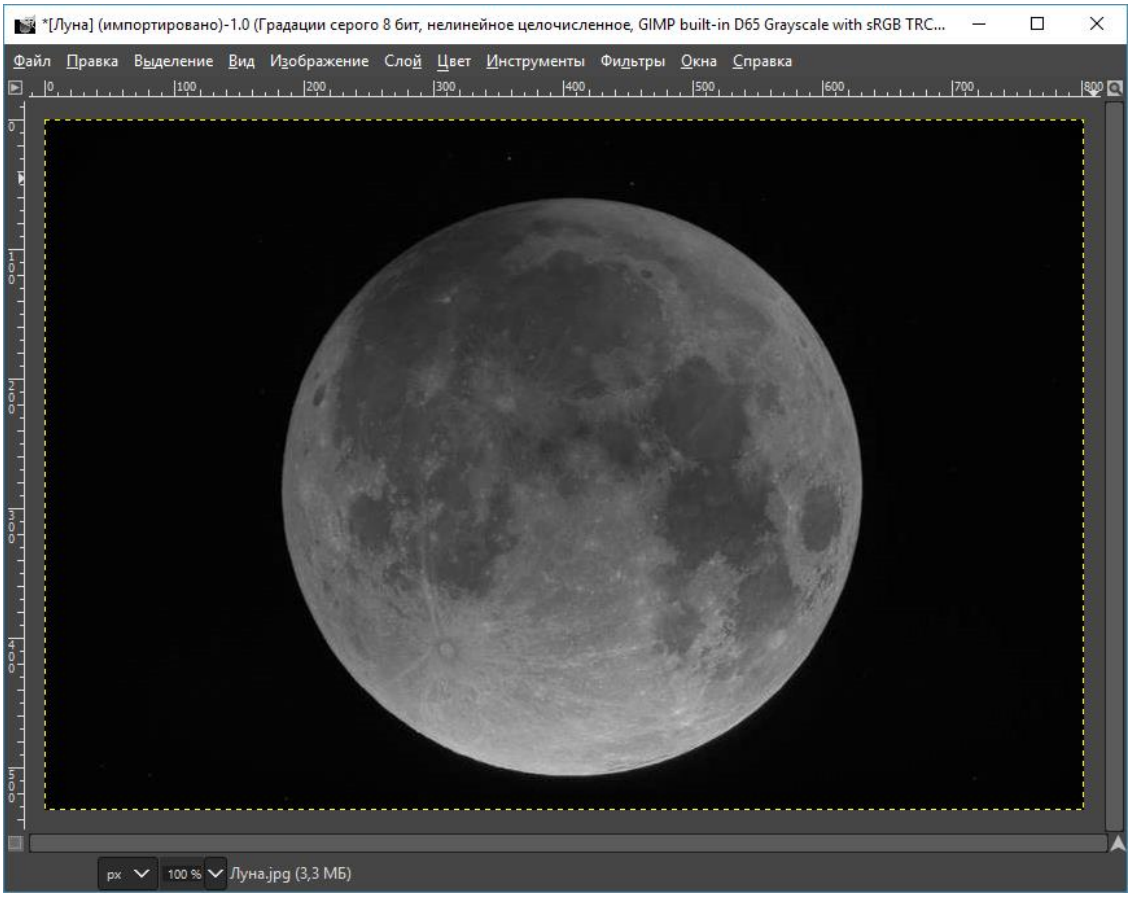

Рисунок 2.24. – Применение режима градации серого в GIMP 2.10.10

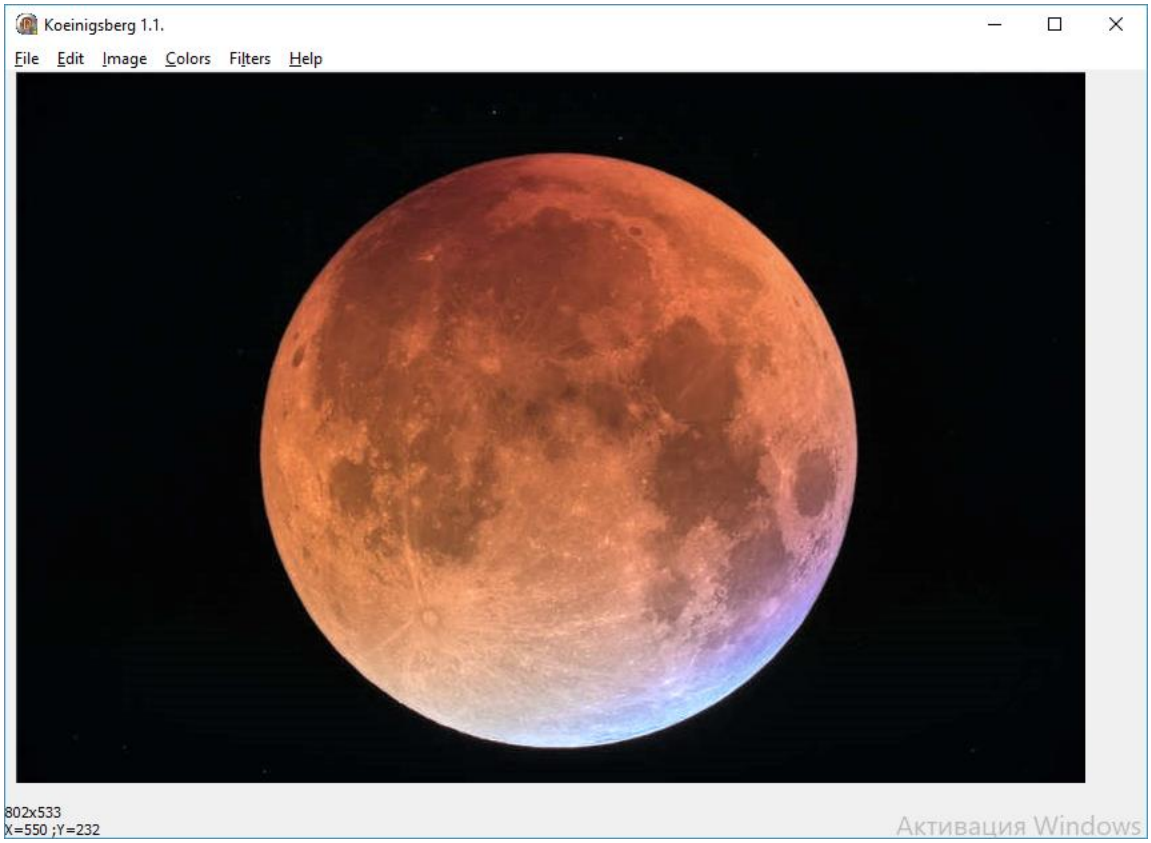

Рисунок 2.25. – Применение линейного контрастирования в Koenigsberg 1.1

2.2.5. Filters

Filters включает в себя разделы (Рисунок 2.26.):

- Noise
- Smoothing filter
- Median filter

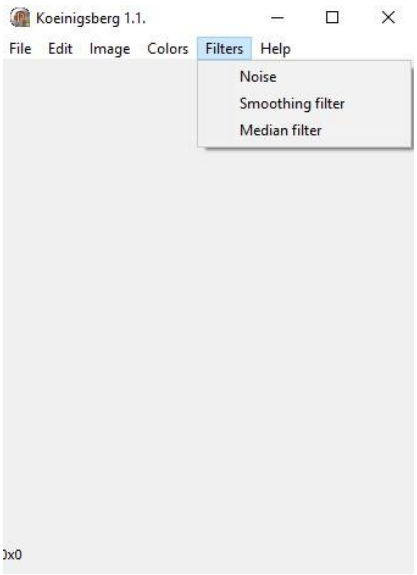

Рисунок 2.26. – Раздел Filters

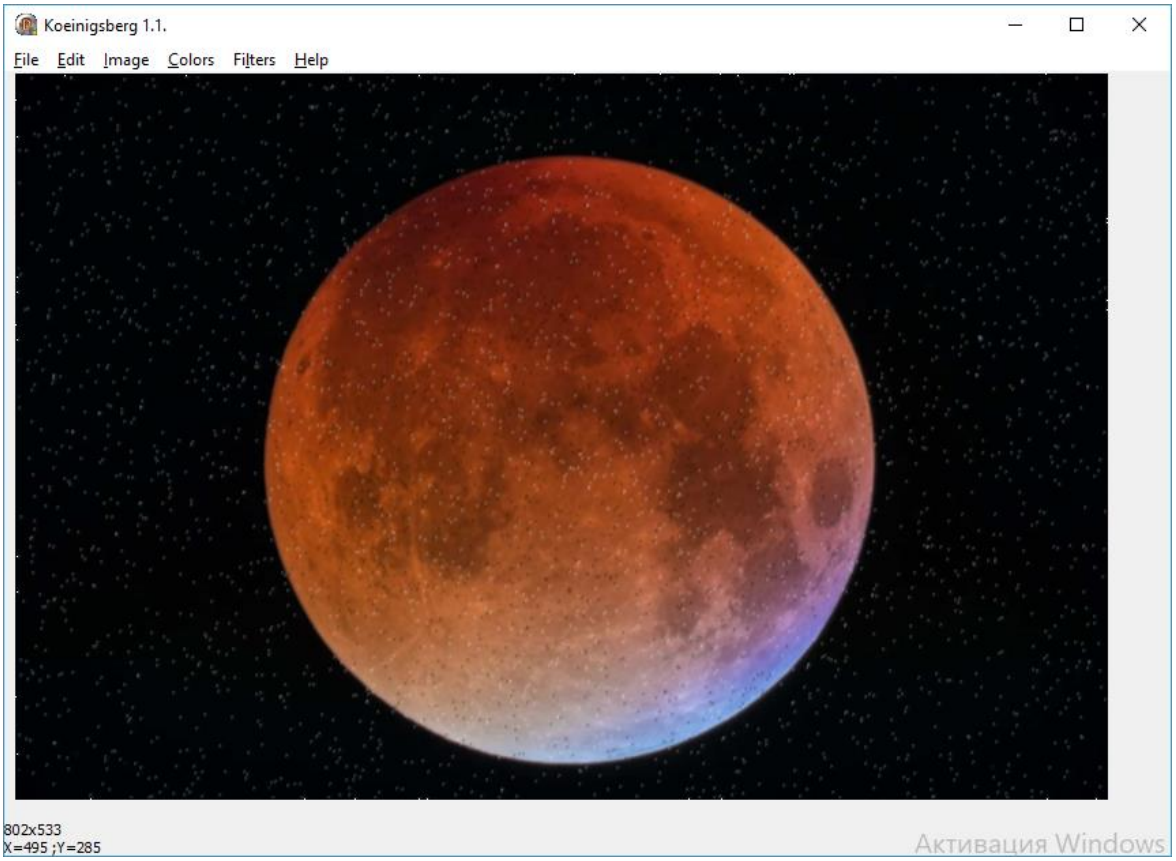

Рисунок 2.27. – Применение сглаживающего фильтра один раз

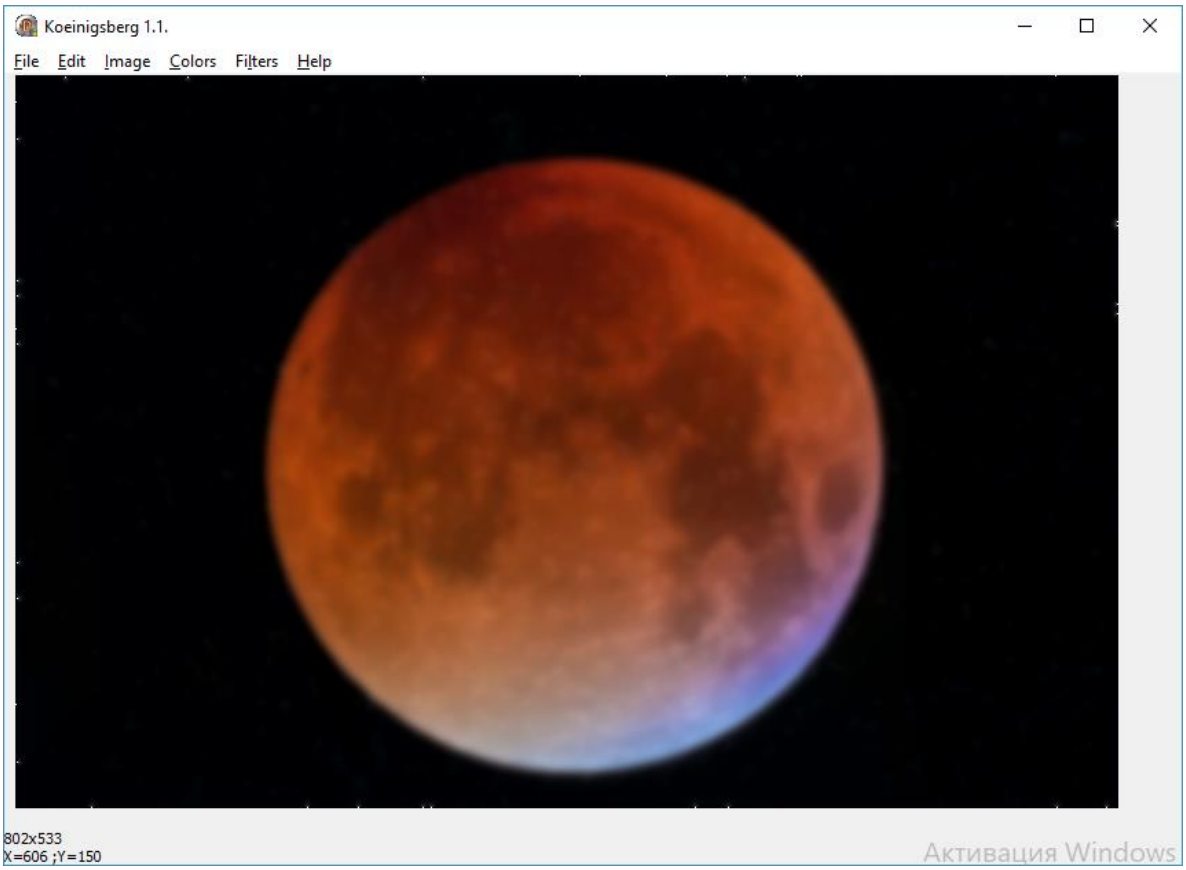

Рисунок 2.28. – Применение сглаживающего фильтра дважды

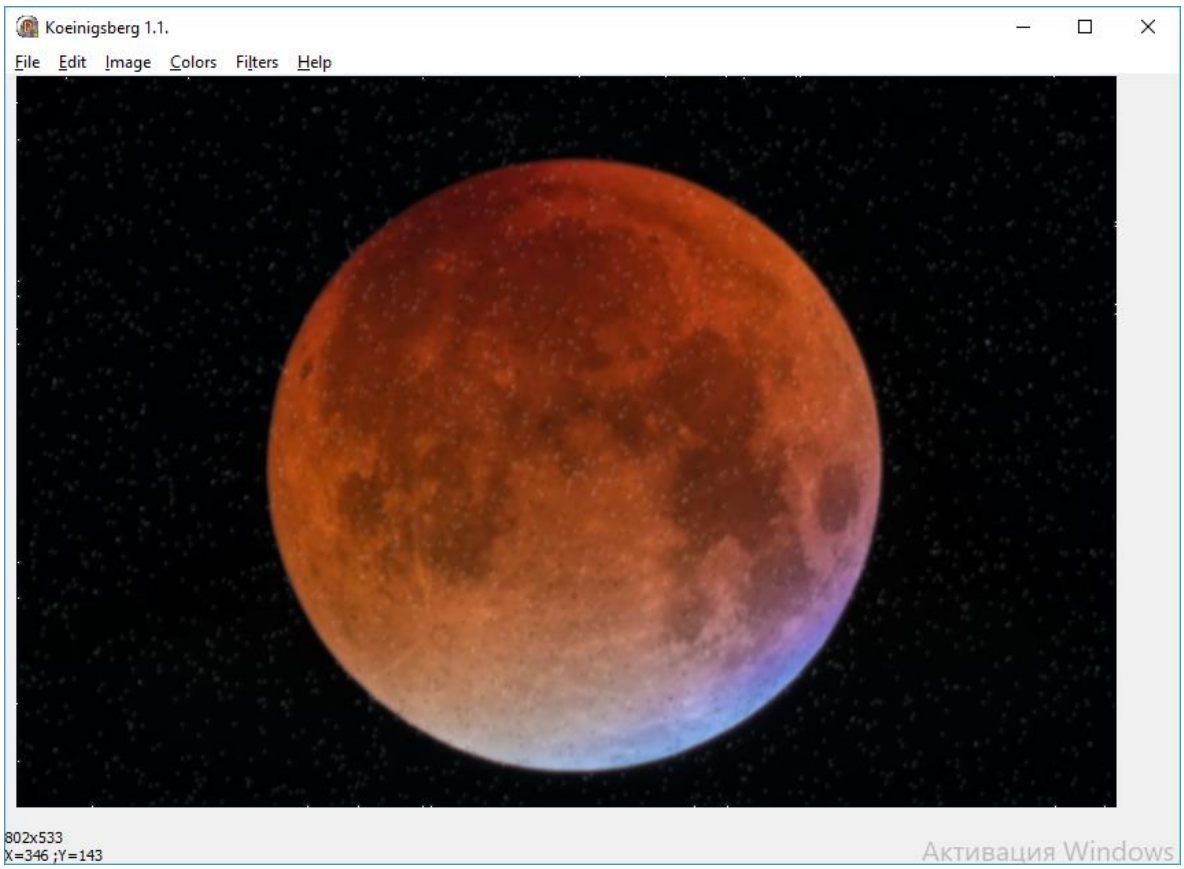

Рисунок 2.29. – Применение сглаживающего фильтра 10 раз в Koenigsberg 1.1

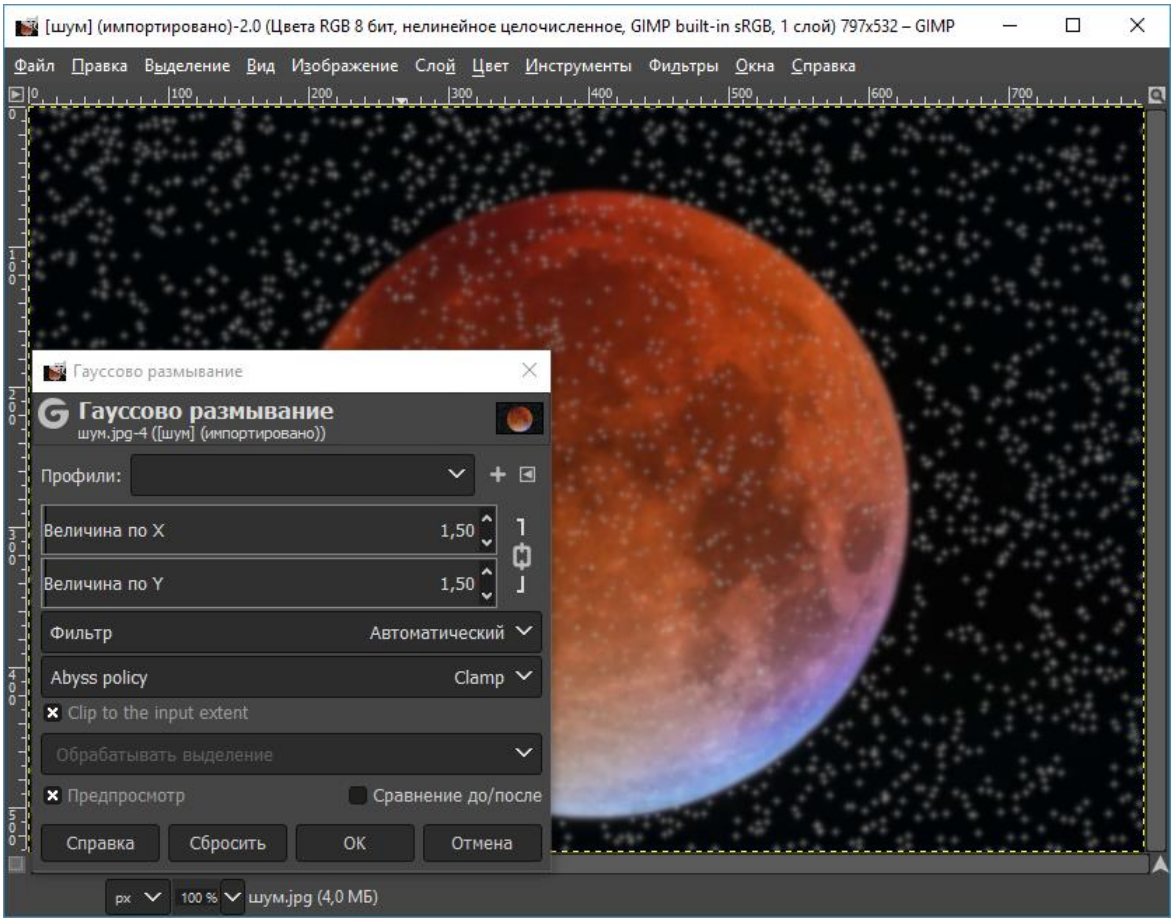

Рисунок 2.30. – Применение гауссово размывание в GIMP 2.10.10

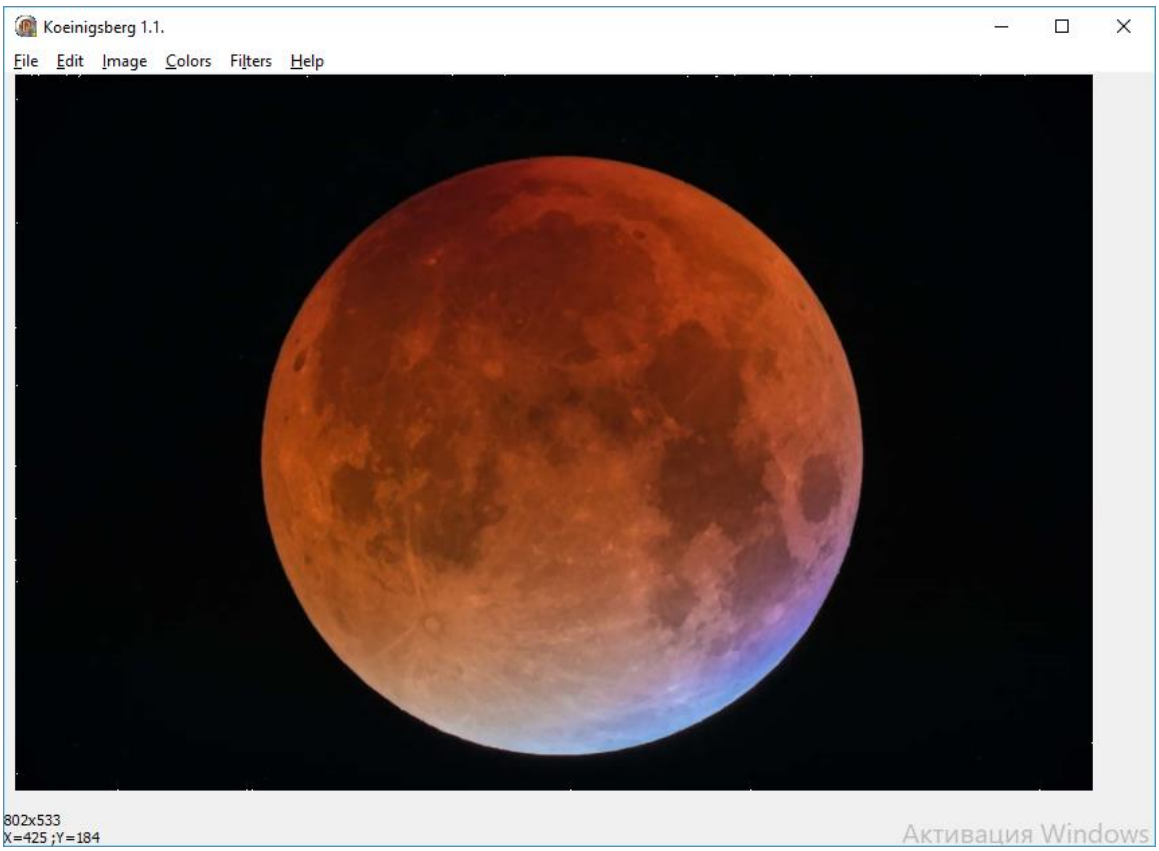

Рисунок 2.31. – Применение медианного фильтра в Koenigsberg 1.1

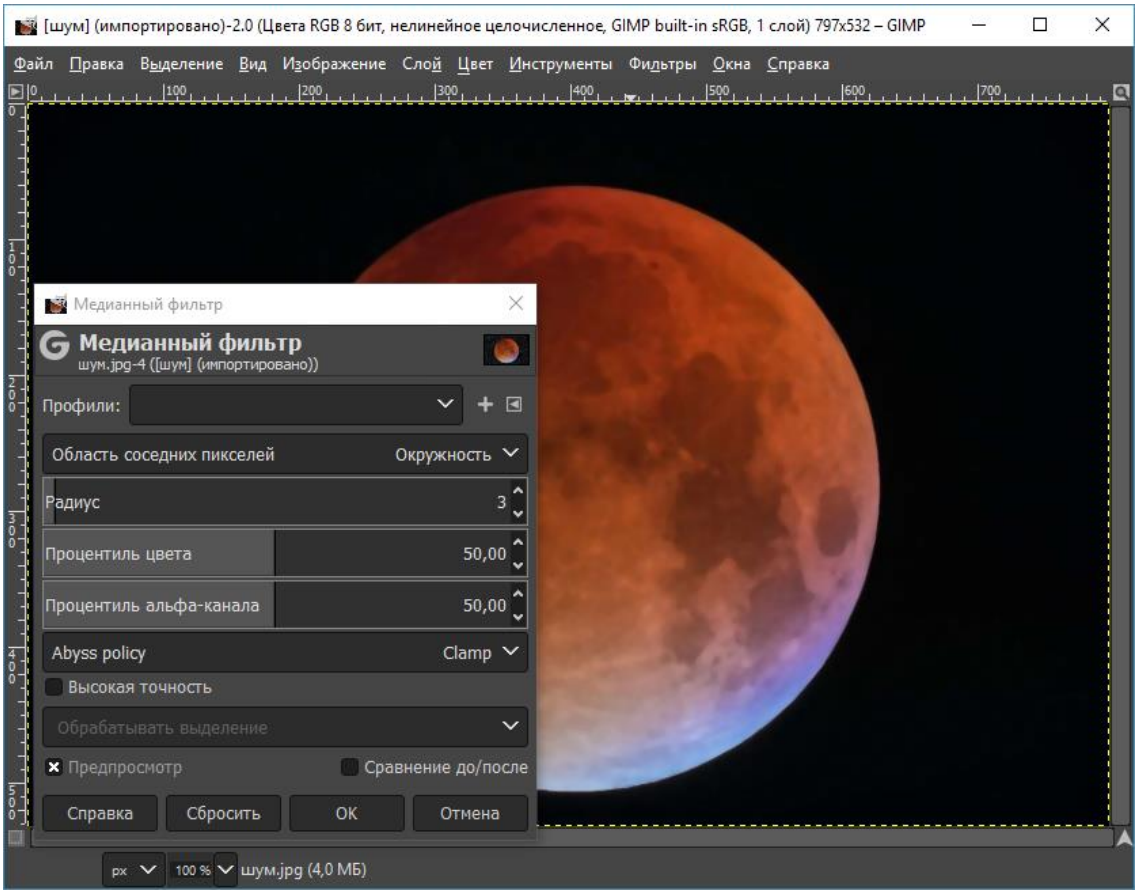

Рисунок 2.32. – Применение медианного фильтра в GIMP 2.10.10

2.2.6. Help

Help включает в себя разделы (Рисунок 2.33.):

- Help
- About

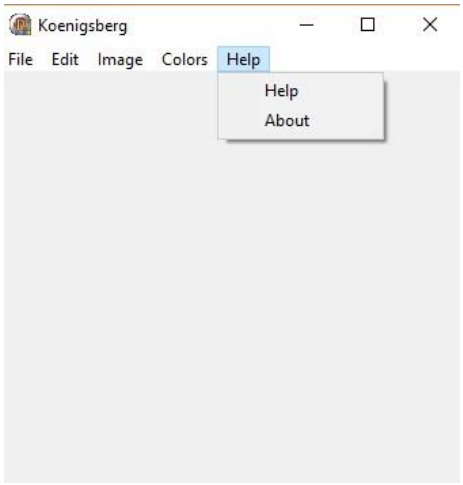

Рисунок 2.33. – Раздел Help

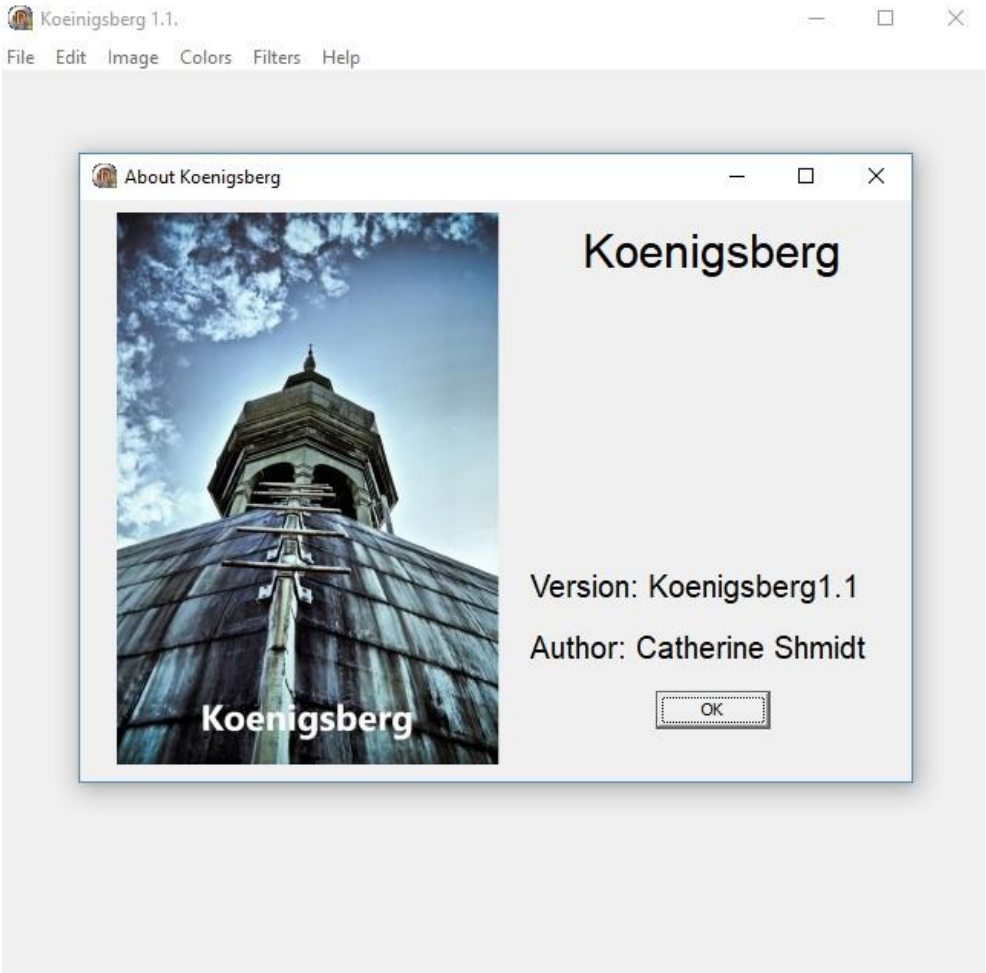

Рисунок 2.34. – Раздел About

Раздел About содержит сведения о программе (Рисунок 2.34.). Также создается новая форма, которая вызывается в OnClick, как немодальное окно. На форме располагаются компоненты Image, три Label и Button (*Button* – простая командная кнопка). В компонент Image загружено изображение, связанное с названием ПО. Загрузка осуществляется следующим образом: при нажатии на свойство Picture появляется окно, содержащее кнопку Load, с помощью которой в Image и будет загружено изображение с жесткого диска. Первый Label в свойстве Caption содержит название создаваемой программы, второй – ее версию, а третий – разработчика данного ПО. Button, названный OK, отвечает за закрытие данной формы. Данная кнопка сделана для удобства пользователя.

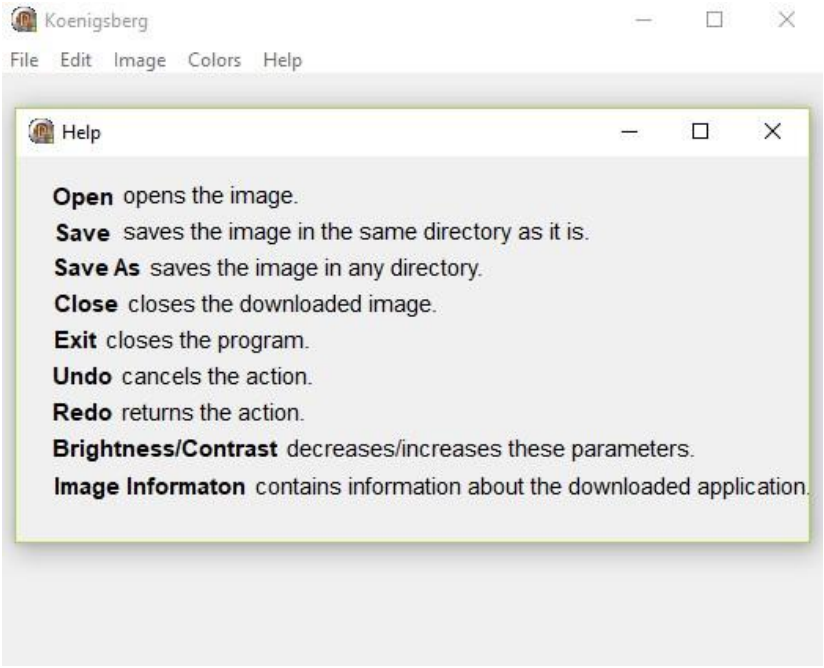

Рисунок 2.35. – Раздел Help

В разделе Help содержится информация, поясняющая все действия, производимые в данном ПО (Рисунок 2.35.). Создается форма, которая вызывается как немодальное окно в обработчике событий OnClick. На форму помещаются компоненты Label. В свойстве Caption[6] данного компонента заносится соответствующая информация.

[6] Caption – свойство, которое используется для задания отображаемого текста у компонента Label.

3. Сравнительный анализ одноименных эффектов/фильтров Koeinigsberg 1.1 с GIMP 2.10.10

В данной работе мы провели сравнительный анализ результатов применения одноименных эффектов/фильтров Koeinigsberg 1.1 с популярным ПО GIMP 2.10.10. Результаты сравнения представлены на рисунках выше (Раздел 2). Как показали результаты режим градации серого, используемый в Koeinigsberg 1.1 визуально совпадает с одноименным режимом в GIMP 2.10.10. после применения их к одному и тому же изображению (Рисунок 2.23, 2.24).

Изображение преобразуется в негатив в GIMP 2.10.10 с помощью вкладки «Инвертировать линейно» (тона и цвета оригинала преобразуются в противоположные). Преобразование приводит к одному и тому же результату (Рисунок 2.21, 2.22).

Для того чтобы получить черно-белым режим в GIMP 2.10.10, необходимо перевести изображение в режим градации серого, а затем определить порог (цвета заменяются либо на черный, либо на белый). Данные режимы незначительно отличаются в обоих ПО, так как используются разные пороги преобразования (Рисунок 2.19, 2.20).

В GIMP 2.10.10 присутствует множество фильтров, накладывающих шум, но не один из них не является фильтром «соль-перец», поэтому в качестве проверки фильтров шумоподавления, проверялось изображение, с наложением шума в ПО Koeinigsberg 1.1.

Линейное контрастирование отсутствует в ПО GIMP 2.10.10.

В GIMP 2.10.10 ни один из фильтров не соответствует сглаживающему фильтру, используемый в Koeinigsberg 1.1.

Медианный фильтр, используемый в Koeinigsberg 1.1 идентичен фильтру с одноименным названием, в котором параметры преобразования стоят по умолчанию, в GIMP 2.10.10 (Рисунок 2.31, 2.32).

Заключение

В данной работе создано ПО Koenigsberg 1.1 с помощью Delphi, в котором реализованы алгоритмы обработки цифровых изображений: негатив, режим градации серого, режим Black and White (черно-белый), изменение яркости и контрастности, построение гистограмм, линейное контрастирование, шум, сглаживающий фильтр, медианный фильтр.

Произведен сравнительный анализ эффектов/фильтров Koeinigsberg 1.1 с популярным GIMP 2.10.10. Результаты показали хорошее внешнее согласие обработанных изображений одноимёнными фильтрами.

Разработанное ПО Koenigsberg 1.1 планируется интегрировать в ПО Вектор-М [1] для обработки комических снимков.

Список используемых источников и литературы

1. V. A. Avdyushev, Banshchikova M.A, I. N. Chuvashov, A. K. Kuzmin Capabilities of software “Vector-M” for a diagnostics of the ionosphere state from auroral emissions images and plasma characteristics from the different orbits as a part of the system of control of space weather // EPSC Abstract, Vol. 11, EPSC2017-834-1, 2017. European Planetary Science Congress 2017. 17–22 September 2017

2. Компьютерная геометрия и графика [Электронный ресурс]// Баньщикова М.А. Учебно-методический комплекс, Томск, 2009. – URL: http://www.astro.tsu.ru/KGaG/ (дата обращения: 28.05.2018).

3. Пиксел. Растр. Характеристики растра [Электронный ресурс]// Студопедия [Б. м.], 2014. – URL: https://studopedia.org/8-205337.html (дата обращения: 28.05.2018).

4. М.Н. Петров, В.П. Молочков. Компьютерная графика. Учебник для вузов. – СПб: «Питер», 2003. – С.736.

5. Что такое негатив изображения? [Электронный ресурс]// Фотошкола Genesis [Б. м.] – URL: http://www.si-foto.com/chto-takoe-negativ-izobrazheniya/ (дата обращения: 28.05.2018).

6. Компьютерная графика :: Яркость [Электронный ресурс]// Уроки Фотошопа [Б. м.] – URL: http://www.lessonsphotoshop.ru/graphics/Index9-2.htm (дата обращения: 28.05.2018).

7. Контрастность изображения [Электронный ресурс]// Фото в нашей жизни [Б. м.] – URL: http://foto-kan.ru/kontrast-v-fotografii/kontrastnost-izobrazheniya.html (дата обращения 28.05.2018).

8. Delphi Sources | Delphi FAQ - Графика и Игры [Электронный ресурс]// Delphi Sources [Б. м.], 2004. – URL:: http://www.delphisources.ru/pages/faq/base/bmp_contrast.html (дата обращения: 24.05.2018).

9. Википедия – свободная энциклопедия [Электронный ресурс] // Гистограмма (фотография) – URL:: https://ru.wikipedia.org/wiki/Гистограмма_(фотография) (дата обращения: 29.03.2019).

10. А. Я. Архангельский. Программирование в Delphi для Windows. Версии 2006, 2007, Turbo Delphi. – М.: ООО «Бином-Пресс», 2007. – С. 1238 (дата обращения: 10.06.2019).

11. Изучаем Delphi – Введение в ООП - SNK Software // SNK Software [Б. м.], 1999. – URL: http://www.snkey.net/books/delphi/ch2-1.html (дата обращения: 27.05.2018).

АНТИПЛАГИАТ
ТВОРИТЕ СОБСТВЕННЫМ УМОМ

Отчет о проверке на заимствования №1

Автор: schmidt-katerina@mail.ru / ID: 6698445
Проверяющий: (schmidt-katerina@mail.ru / ID: 6698445)

Отчет предоставлен сервисом «Антиплагиат»- http://users.antiplagiat.ru

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 6
Начало загрузки: 13.06.2019 11:33:53
Длительность загрузки: 00:00:04
Имя исходного файла: Diplom
Размер текста: 3310 кБ
Символов в тексте: 40121
Слов в тексте: 4682
Число предложений: 339

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Последний готовый отчет (ред.)
Начало проверки: 13.06.2019 11:33:58
Длительность проверки: 00:00:02
Комментарии: не указано
Модули поиска: Модуль поиска Интернет

ЗАИМСТВОВАНИЯ	ЦИТИРОВАНИЯ	ОРИГИНАЛЬНОСТЬ
21,34%	0%	78,66%

Заимствования — доля всех найденных текстовых пересечений, за исключением тех, которые система отнесла к цитированиям, по отношению к общему объему документа.
Цитирования — доля текстовых пересечений, которые не являются авторскими, но система посчитала их использование корректным, по отношению к общему объему документа. Сюда относятся оформленные по ГОСТу цитаты; общеупотребительные выражения; фрагменты текста, найденные в источниках из коллекций нормативно-правовой документации.
Текстовое пересечение — фрагмент текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника.
Источник — документ, проиндексированный в системе и содержащийся в модуле поиска, по которому проводится проверка.
Оригинальность — доля фрагментов текста проверяемого документа, не обнаруженных ни в одном источнике, по которым шла проверка, по отношению к общему объему документа.
Заимствования, цитирования и оригинальность являются отдельными показателями и в сумме дают 100%, что соответствует всему тексту проверяемого документа.
Обращаем Ваше внимание, что система находит текстовые пересечения проверяемого документа с проиндексированными в системе текстовыми источниками. При этом система является вспомогательным инструментом, определение корректности и правомерности заимствований или цитирований, а также авторства текстовых фрагментов проверяемого документа остается в компетенции проверяющего.

№	Доля в отчете	Источник	Ссылка	Актуален на	Модуль поиска
[01]	5,95%	Огрызков С А Курсовая работа по обработке изображений	http://masters.donntu.org	15 Окт 2018	Модуль поиска Интернет
[02]	0%	Яркость точки и гистограммы изображения	http://rerefat.ru	23 Мар 2016	Модуль поиска Интернет
[03]	2,32%	Программирование на Delphi	http://decoding.dax.ru	23 Авг 2017	Модуль поиска Интернет

Еще источников: 12
Еще заимствований: 13,07%