Лабораторная работа №2
Фильтры

Титов Виктор Викторович
e-mail: vtitov@rtc.ru

Цель: изучить базовые алгоритмы фильтрации изображений в пространственной и реализация фильтров.

Задачи:
1) Изучить основные виды фильтров
2) Ознакомиться с базовыми функциями фильтрации в OpenCV
3) Проверить работу фильтров на практике
4) Выполнить практическое задание в конце

ТЕОРИЯ

Фильтрация в пространственной может рассматриваться, как некоторый оператор действующий на окрестность точки (x,y) на изображении

$$g(x,y) = T[f(x,y)]$$

$$T: R^{mxn} \rightarrow R$$

f – исходное изображение размером NxM,
T– оператор, действующий на область mxn изображения f в окрестности точки (x,y)
g– результирующее фильтрованное изображение

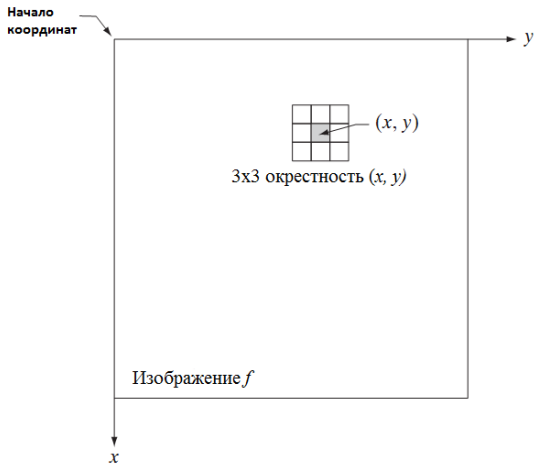


Рисунок 1 – Фильтрация в пространственной области

Преобразование интенсивности

Одной из базовых форм операторов, действующих на изображение является преобразование интенсивности. Данные формы преобразования имеют (как правило) локальную область в один пиксель (n=m=1) и предназначены для коррекции интенсивности пикселей в независимости от их локальной области.

К таким преобразованиям можно отнести:

1. Изменение средней яркости изображения – добавление фиксированного значения яркости к каждому пикселю

$$g(x,y) = f(x,y) + \mathrm{c}$$

c – постоянный сдвиг интенсивности

В OpenCV данную процедуру можно реализовать с помощью простой операции сложения

M += Mat::eye(M.rows, M.cols, CV_64F);

2. Нормализация интенсивности изображения – процедура, при которой интенсивность каждого пикселя умножается на константу, приводящее к масштабированию интенсивностей к максимальному диапазону формата представления данных

$$g(x,y) = f(x,y) \cdot 255/\max f$$

Схожий метод нормировки предусматривает также обнуление самого маленького значения интенсивности пикселя на изображении

$$g(x,y) = f(x,y) \cdot \frac{255}{\max f - \min f} - minf$$

3. (разновидность) Негатив – интенсивности пикселей изображения обращаются относительно диапазона представления данных

$$g(x,y) = 255 - f(x,y)$$

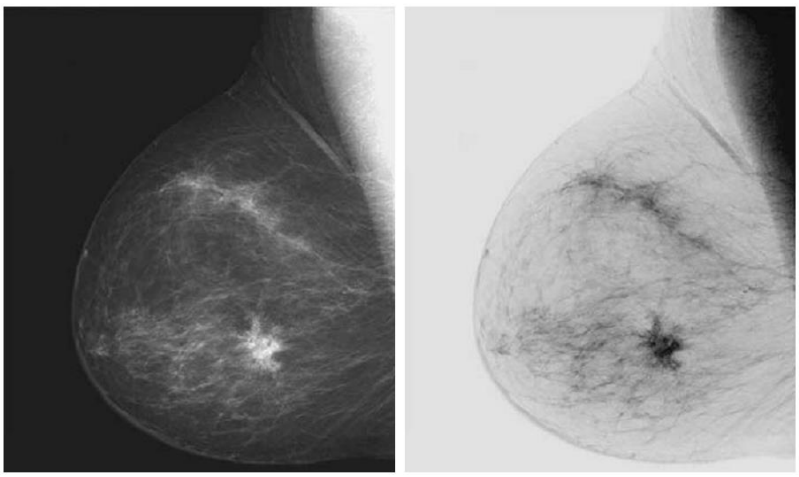


Рисунок X – Пример негативного преобразования изображения (момограмма)

Применяется для улучшения восприятия человеком деталей изображения.

4. Логарифмическое представление

$$g(x,y) = \mathrm{c} \cdot \log\big(1 + f(x,y)\big)$$

Логарифмическое представление применяется в тех случаях, когда интенсивности на изображении имеют большой разброс (присутствуют как очень светлые так и совсем темные пиксели), а также когда интенсивности меняются слишком быстро/медленно, чтобы можно было заметить тренд. В этом случае логарифмическое представление делает изображение более «читаемым» для человека. В OpenCV реализована функция

cv::intensity_transform::logTransform

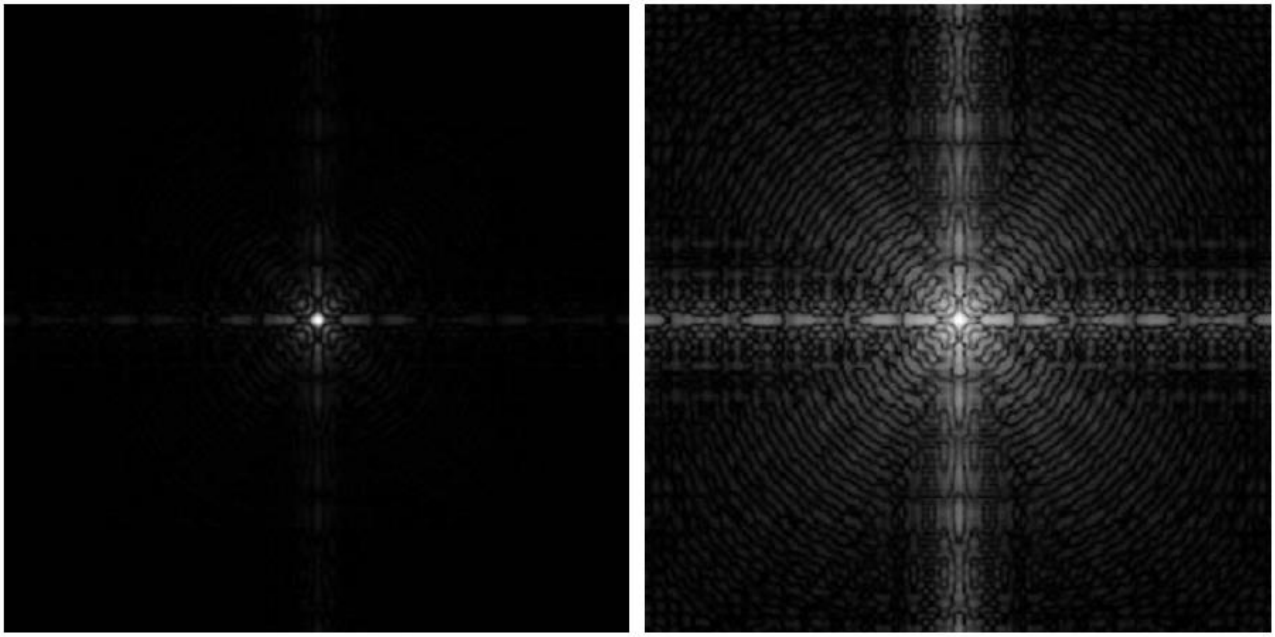


Рисунок X – Пример логарифмического преобразования

5. Гамма коррекция

$$g(x,y) = \mathrm{c} \cdot (f(x,y) + \epsilon)^{\gamma}$$
$$\gamma, c = const > 0$$

ϵ– малая величина, если необходим не нулевое значение на выходе при нулевом входе

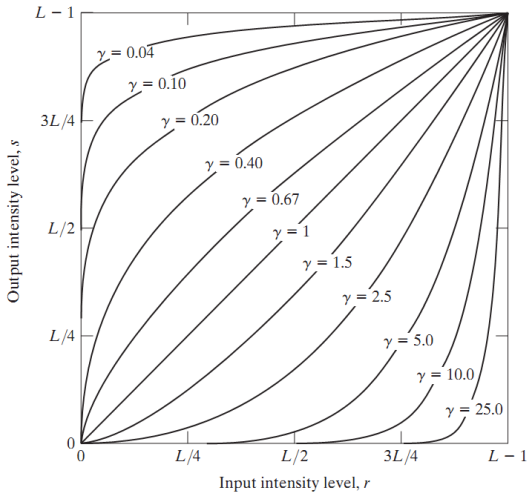


Рисунок 2 – Функция гамма преобразования интенсивности

Многие из приборов имеют экспоненциальную характеристику от измеряемой величины (например, МРТ аппараты, рентген в сочетании со считывающей матрице, др). Гамма коррекция позволяет правильно отобразить их на экране монитора.

Функция OpenCV: **cv::intensity_transform::gammaCorrection**

Рисунок – Справа исходное изображение, слева – после гамма преобразования

6. Общий случай коррекции интенсивности

$$g(x,y) = G(f(x,y))$$

G– биективная (взаимно-однозначная) функция интенсивности

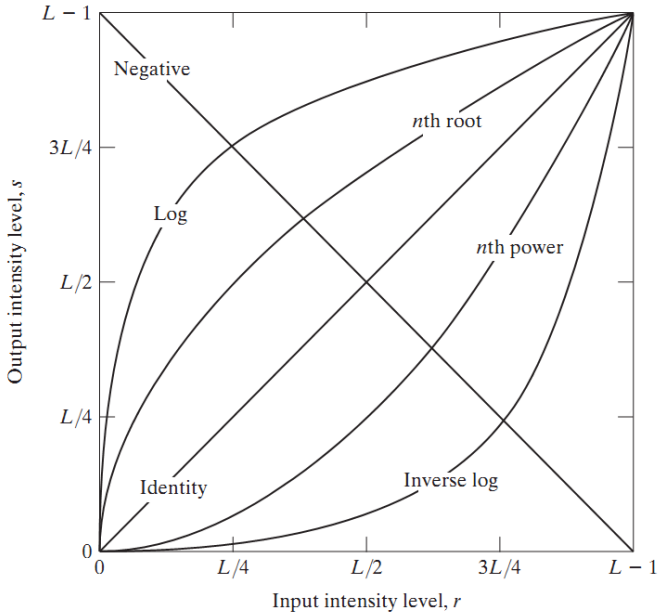


Рисунок 2 – функции преобразования интенсивности

Заметим, что функция преобразования интенсивности должна быть биективной (т.е. взаимно-однозначным преобразованием, имеющим обратное преобразование). Иначе возникнет потеря данных, т.е. одни пиксели окажутся той же интенсивности, что другие и они станут неразличимы.

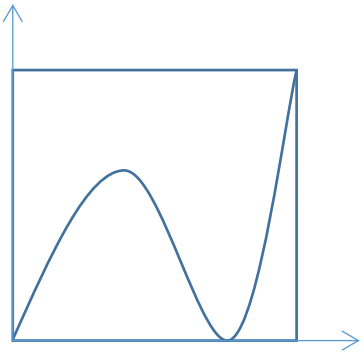


Рисунок 3 – Пример не биективного преобразования

Чисто теоретически такое преобразование допустимо, но необходимо понимание в каждом конкретном случае, как и зачем его применять.

7. Фильтрация по уровню интенсивности – на выходе данного фильтра остаются только те пиксели изображения, интенсивность которых попадает в заданный диапазон

$$g(x,y) = \begin{cases} f(x,y), level_{max} \le f(x,y) \le level\_max \\ 0 \end{cases}$$

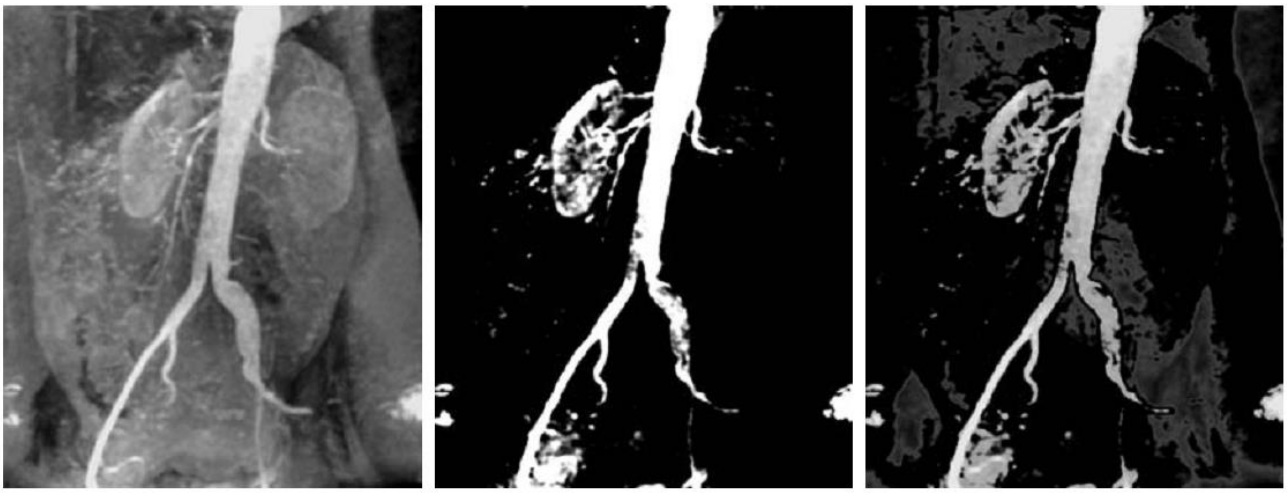


Рисунок – Пример фильтрации изображения по уровню (выделение сосудов на снимке)

Основное применение данного фильтра – выделение областей на изображении, отвечающих за объекты с определённым уровнем поглощения излучения (рентгенографические исследования, материаловедение, т.д.).

Функция OpenCV: cv::inRange

8. Bit slicing – разновидность фильтрации по уровню интенсивности, когда за уровень принимается какой-то бит в двоичном представлении. Для 8-ми битного изображения каждый бит отвечает некоторому слою изображения

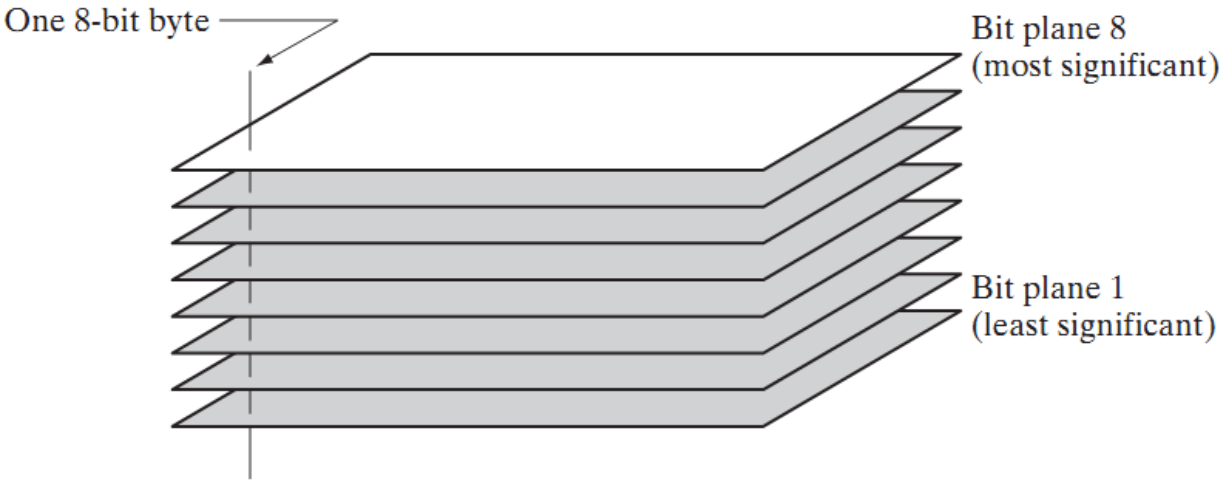


Рисунок – Побитовые слои изображения

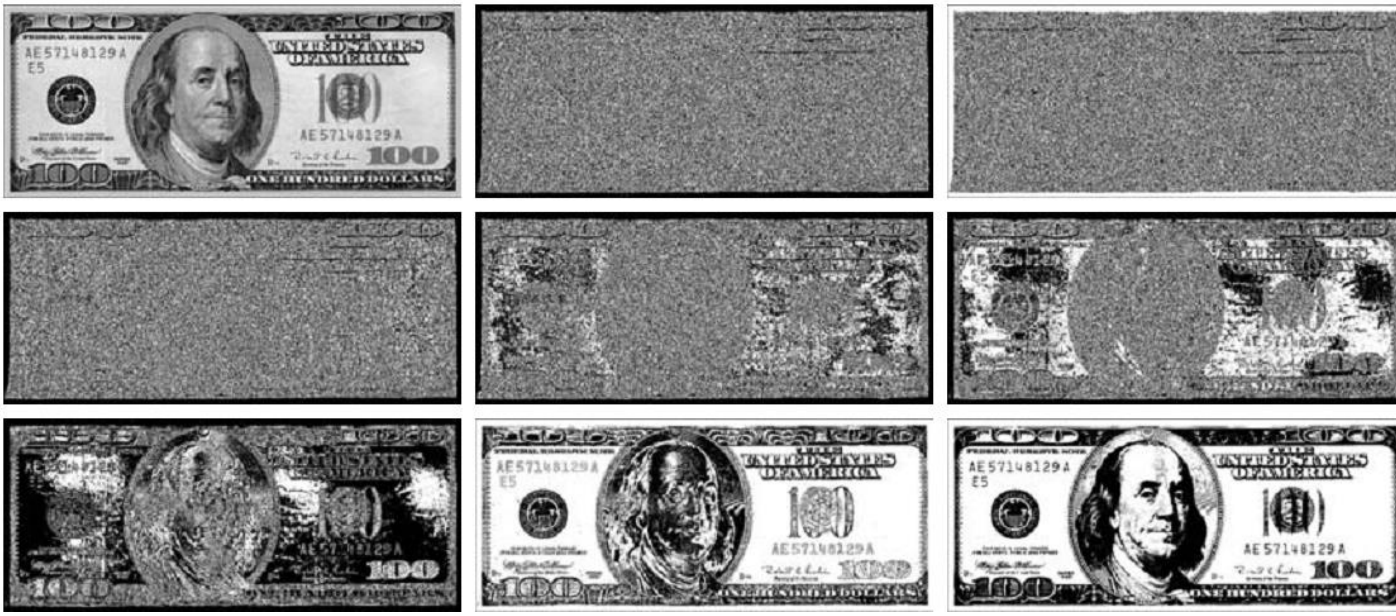


Рисунок – пример применения bit slicing для банкноты. Левое верхнее изображение – оригинал. Слева на право, сверху вниз биты 0-7

9. Гистограммная коррекция – название происходит от формы представления данных в статистике.

Гистограмма – график, представленный вертикальными столбцами, высота которых пропорциональная числу элементов, попадающих в диапазон величин, откладываемый по оси абсцисс. В статистике гистограмма используется для отображения эмпирически полученной функции распределения.

Для изображения гистограмма отражает распределение интенсивностей на изображении, т.е. какое число пикселей с каждым уровнем интенсивности присутствует на изображении.

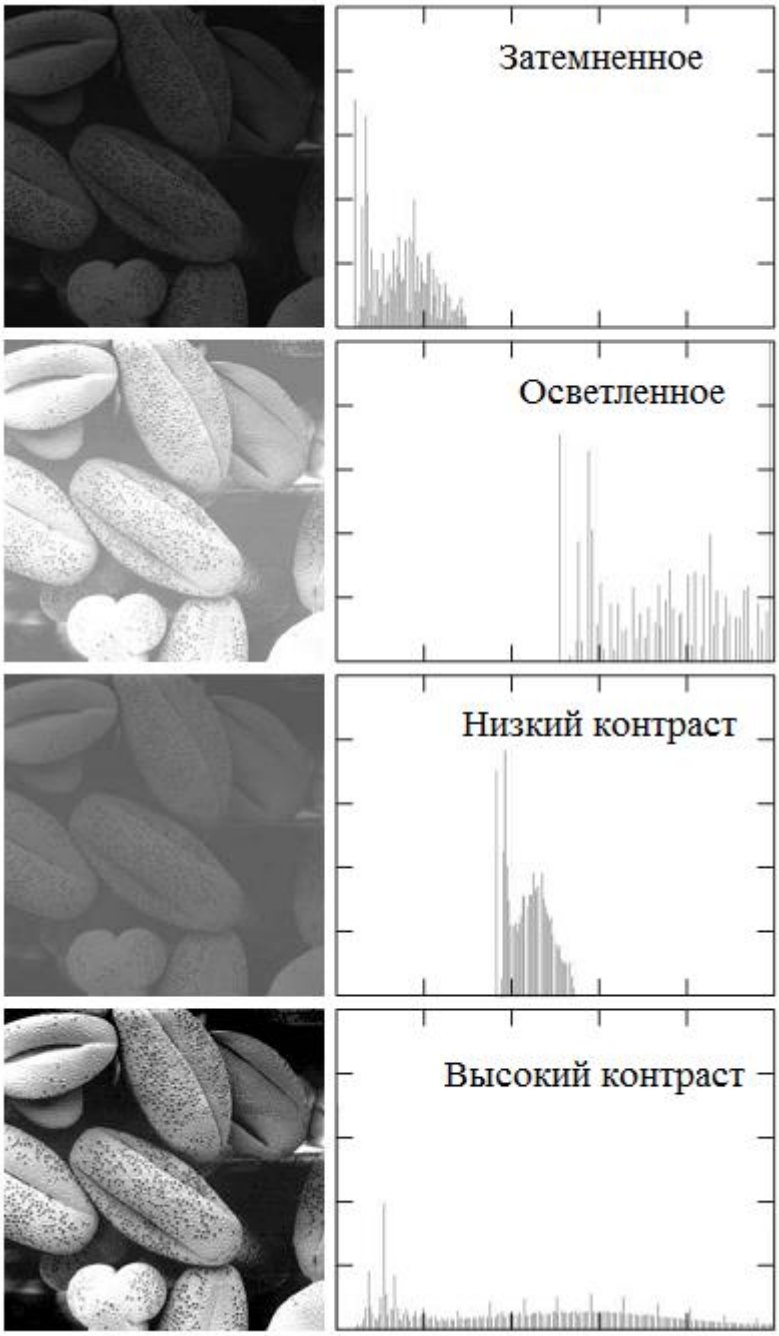


Рисунок – Пример гистограммы grayscale изображения

Как видно из рис. Изображение с хорошим контрастом имеет почти равномерное распределение интенсивностей. Поэтому гистограммная коррекция дает улучшение контрастности на изображении и делает своеобразную нормировку.

Интенсивность пикселя на изображении можно рассматривать, как случайную переменную, характеризуемую плотностью распределения.

Пусть имеется функция преобразования интенсивности

$$g(x, y) = H(f(x, y))$$

H– монотонная гладкая неубывающая функция (неубывающая, т.к. нужно чтобы более светлые писксели не становились темнее более темных после преобразования)

Путь имеется две плотности вероятности $p_g(g(x, y))$ и $p_f(f(x, y))$ для преобразованного и исходного изображений. Т.к. функция H является неубывающей, то справедливо соотношение

$$\int_0^{g=H(f)} p_g(w)dw = \int_0^f p_f(s)ds$$

Производя замену переменных

$$\int_0^f p_g(H(f))dH(f) = \int_0^f p_f(s)ds$$

$$\int_0^f p_g(H(s))\frac{dH(s)}{ds}ds = \int_0^f p_f(s)ds$$

$$p_g(H(s))\frac{dH(s)}{ds} = p_f(s)$$

или без преобразования

$$p_g(w) = p_f(s)\cdot\left.\frac{ds}{dw}\right|_{w=H(s)}$$

Заметим, что

$$w = H(s)$$

$$\frac{dw}{ds} = \frac{dH(s)}{ds}$$

Потребуем, чтобы распределение преобразованного изображения было равномерным

$$p_g(w) = \frac{1}{L},\ 0 \le \mathrm{s} \le L$$

Тогда

$$p_f(s)\cdot\left(\frac{dH(s)}{ds}\right)^{-1} = \frac{1}{L-1}$$

Откуда мы получаем дифференциал от преобразования

$$\frac{dH(s)}{ds} = (L-1)\cdot p_f(s)$$

L -1 – нормирующий фактор (равный максимальной интенсивности на изображении)

Откуда

$$w = H(s) = \int_0^s \frac{dH(\mathrm{t})}{dt}dt = (L-1)\cdot\int_0^s p_f(t)\,\mathrm{dt}$$

Или в дискретной форме

$$w_k = H(s_k) = (L-1)\cdot\sum_{i=0}^{k} p_f(\mathrm{s}_i)$$

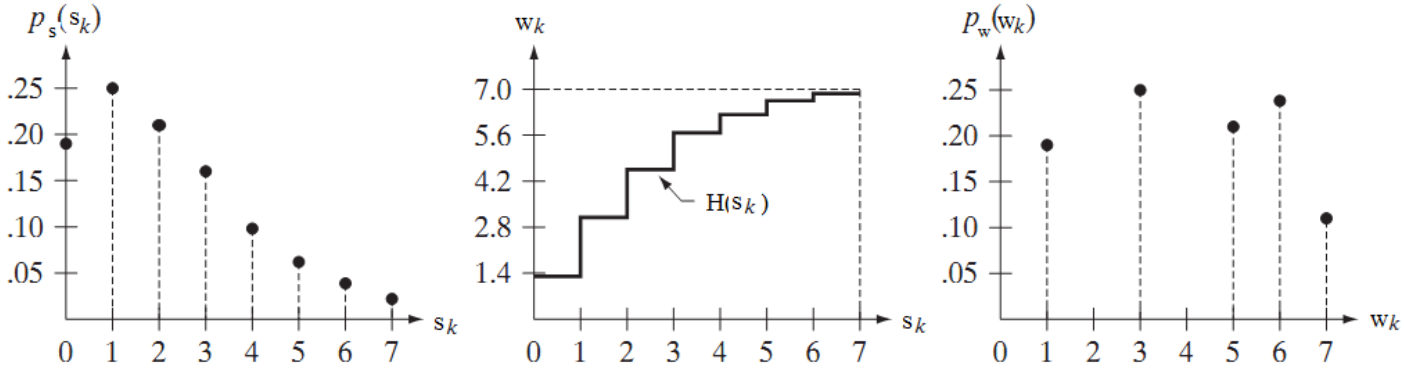


Рисунок – иллюстрация выравнивания гистограммы

Фильтрация в пространственной области

Основой фильтрации в пространственной области является выбор значения интенсивности пикселя на основе значений интенсивностей пикселей в его окрестности.

В основе линейной фильтрации лежит операция свертки

$$g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x + s, y + t)$$

$w(.,.)$ – ядро свертки или оконная функция

Операцию свертки можно интерпретировать, как получения значения интенсивности в точке (x,y) как взвешенное среднее от ее окружения с весами $w(.,.)$.

В общем случае функция весов должна быть нормирована: сумма ее элементов должна быть равна 1.

$$\sum_{i=-a}^{a} \sum_{j=-b}^{b} w(i, j) = 1$$

В абстрактном смысле это можно понимать, как то, что фильтр не должен изменять общую энергию фильруемой области, а должен только перераспределять ее. В более простом случае можно рассуждать так. Фильтр, фильтрующий монотонную область изображения не должен изменять интенсивности пикселей в ней.

$$\sum_{i=-a}^{a} \sum_{j=-b}^{b} w(i, j) \cdot f(x + i, y + j) = g(x, y)$$

$$f(.,.) = g(.,.) = \mathrm{c} = \mathrm{const}$$

$$\sum_{i=-a}^{a} \sum_{j=-b}^{b} w(i, j) \cdot c = c \Rightarrow \sum_{i=-a}^{a} \sum_{j=-b}^{b} w(i, j) = 1$$

Поэтому формулу фильтров часто представляют в нормированном виде

$$g(x, y) = \frac{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x + s, y + t)}{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t)}$$

Сглаживающие фильтры

Шумы на изображении, как правило представлены резкими случайными изменениями интенсивности на изображении. Распределение таких изменений определяет тип шума. Такие резкие изменения являются негативным фактором, т.к. искажают локально информацию на изображении.

Уменьшить такой шум позволяют сглаживающие фильтры, убирающие резкие изменения (перепады) интенсивности на изображении.

Рассмотрим два основных фильтра, применяемых в техническом зрении

1. Box filter или фильтр среднего

Применяется для быстрой, но низкокачественной фильтрации изображений. Представляет собой взвешенной среднее пикселей на изображении.

$\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

$\frac{1}{16} \times$

1	2	1
2	4	2
1	2	1

Рисунок – Примеры ядра фильтров среднего

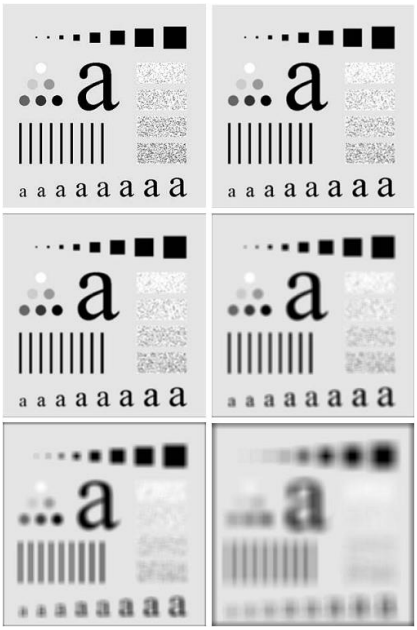


Рисунок – Примеры работы фильтра среднего (box filter)

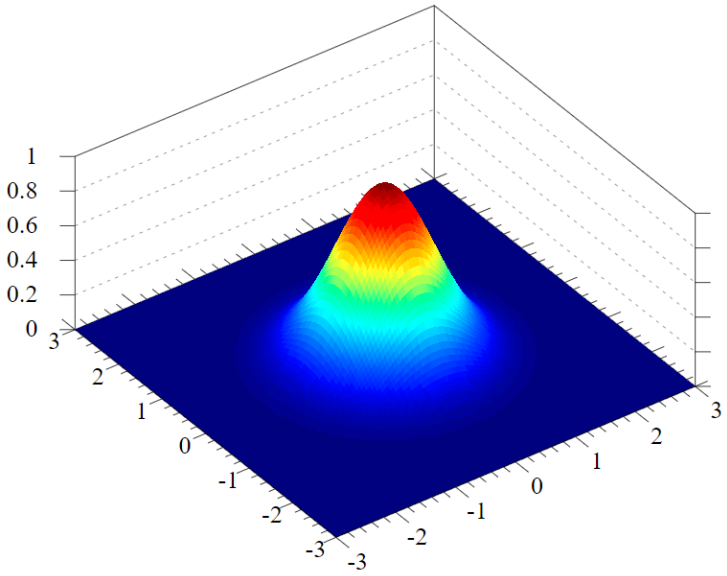
В простейшем случае box filter имеет формулу

$$g(x,y)=\frac{1}{n\cdot m}\sum_{i=-(n-1)/2}^{(n-1)/2}\sum_{j=-(m-1)/2}^{(m-1)/2}f(x+i,y+j)$$

Функция OpenCV: **cv::blur**

2. Гауссов фильтр – фильтр ядром которого является двумерный гессиан

$$w(i,j)=e^{-\frac{i^2+j^2}{2\sigma}}$$



Сам процесс фильтрации представлен нормированной формулой выше.

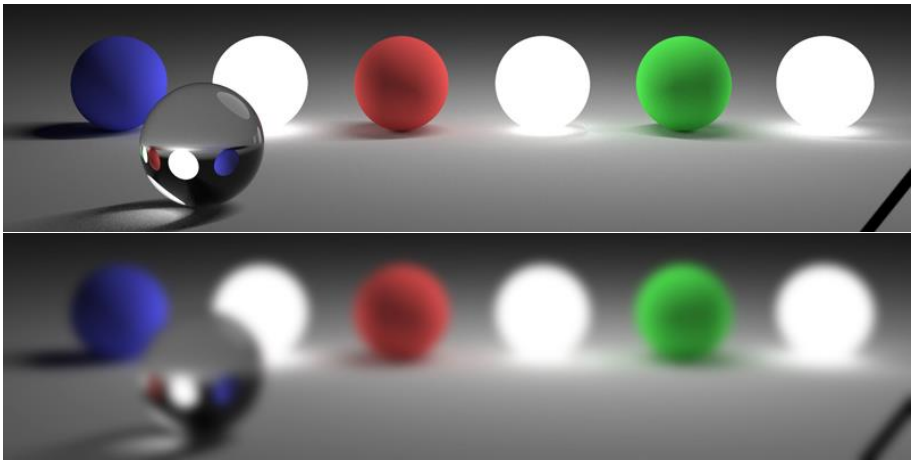


Рисунок – пример гауссовой фильтрации

Функция OpenCV: **cv::GaussianBlur**

3. Фильтры основанные на статистике. Группа фильтров, назначающая значение интенсивности пикселей на изображении на основе статистических показателей.

Наиболее распространённым является медианный фильтр

Медиана набора чисел — число, которое находится в середине этого набора, если его упорядочить по возрастанию, то есть такое число, что половина из элементов набора не меньше него, а другая половина не больше.

Медиана набора чисел — это число, сумма расстояний (или, если более строго, модулей) от которого до всех чисел из набора минимальна. [https://en.wikipedia.org/wiki/Median]

В данном наборе
1, 3, 3, **6**, 7, 8, 9
медиана равна 6
А в данном 1, 2, 3, **4, 5**, 6, 8, 9 – медиана равна 4,5

Медианный фильтр применяется в тех случаях, когда шум на изображении представлен нерегулярными выбросами (как правило в величину всего диапазона измерений).

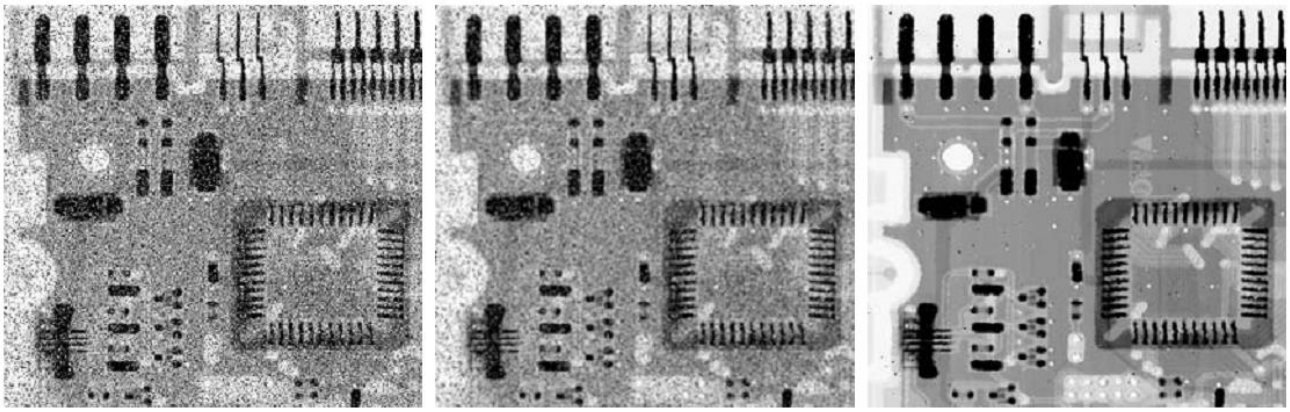


Рисунок – Пример работы фильтров, слева на право: исходное изображение, фильтр среднего, медианный фильтр

Функция OpenCV: **cv::medianBlur**

Фильтры повышения резкости

Основные требования к функции улучшения резкости

- Не изменяет области с постоянной интенсивностью
- Не изменять пространственного расположения границ
- В местах изменения интенсивности увеличивать скорость ее изменения (создавать резкий переход)

Рассмотрим первую и вторую производные для функции дискретной переменной

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

И рассмотрим произвольную функцию и ее первую и втору производные

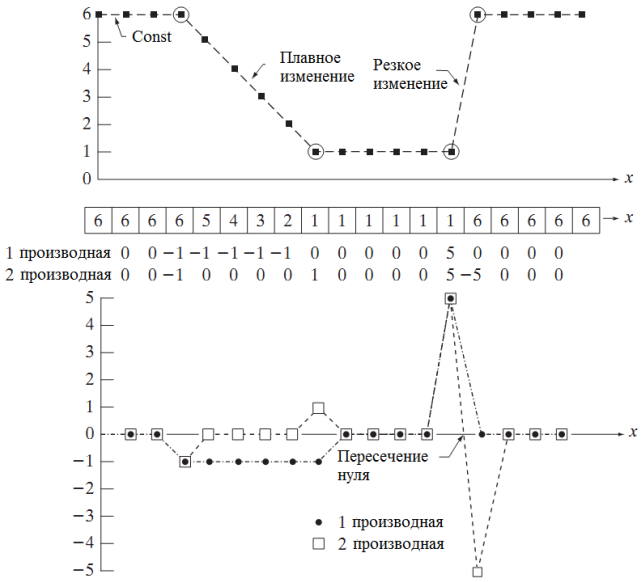


Рисунок – Дискретная функция и ее первая и вторая производные

Из изображения мы видим, что вторая производная обладает необходимыми данными для повышения резкости изображения. Она показывает те места, где изменение интенсивности пикселя должно быть подчеркнуто, т.е. в начале и в конце областей измененбия интенсивностей.

Лапласианом называется сумма вторых производных от двумерной (многомерной) функции.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

В дискретном виде

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

$$\nabla^2 f(x, y) = f(x + 1, y) + f(x - 1, y) + f(x, y + 1) + f(x, y - 1) - 4f(x, y)$$

Заметим, что Лапласиан состоит из линейным операция и, т.о., является линейным фильтром

0	1	0	1	1	1
1	−4	1	1	−8	1
0	1	0	1	1	1
0	−1	0	−1	−1	−1
−1	4	−1	−1	8	−1
0	−1	0	−1	−1	−1

Рисунок – Варианты ядра Лапласиана

Увеличить контраст на изображении можно вычитая Лапласиан изображения из исходного изображения с некоторым коэффициентом

$$g(x, y) = f(x, y) + c\left[\nabla^2 f(x, y)\right]$$

Коэффициент c<0, выбирается так, чтобы достичь желаемого улучшения резкости.

Более дешевой реализацией улучшения резкости на изображении является применение метода unsharp masking. Он заключается в последовательности операций:

1) Сгладить изображение сглаживающим фильтром
2) Вычесть сглаженное изображение из исходного

$$g_{\text{mask}}(x, y) = f(x, y) - \overline{f}(x, y)$$

$\overline{f}$ – сглаженное изображение

3) Добавить получившуюся разность к исходному изображению

$$g(x, y) = f(x, y) + k * g_{\text{mask}}(x, y)$$

k – коэффициент, определяющий степень увеличения резкости

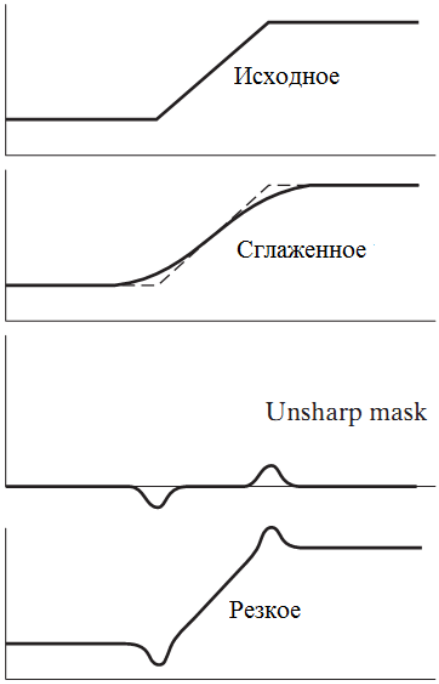


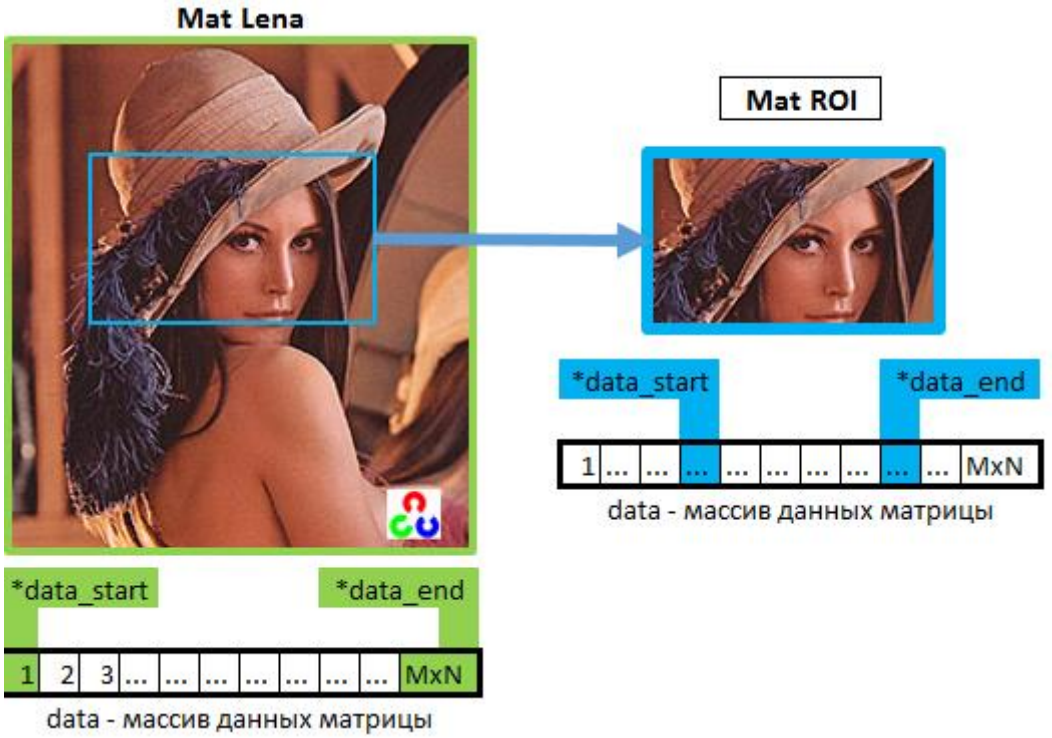
Рисунок – Работа unsharp masking

ПРАКТИЧЕСКАЯ РАБОТА

1. Для написания собственных фильтров в OpenCV удобно использовать такую вещь как *ROI(Region of Interest)*, а если по-нашему, то *Область Интереса(ОИ)*.

Данная возможность позволяет выделить область исходного изображения и работать с данной областью как с отдельным изображением, при этом все изменения на выделенной области будут переноситься на исходное изображение, т.к. ОИ не копирует содержимое, а указывает (см. в поисковике «указатели с/с++») на него. Это позволяет меньше заботиться о преобразовании координат при создании собственного фильтра или какого-нибудь алгоритма поиска по типу скользящего окна.

Для иллюстрации принципа работы используем известное изображение *lena.jpg* (https://ru.wikipedia.org/wiki/Лена_(изображение)).



Как мы видим массив данных остаётся тот же, но меняются указатели на его начало и конец. (А что такое указатели, что за волшебство? Если не знаешь, то почитай или забей, если лень. Но лучше не забивай, т.к. без них тебе будет трудновато написать хорошую программу на C/C++, чтобы тебя потом не материли пользователи).

Теперь же приступим к заданию ROI в OpenCV:

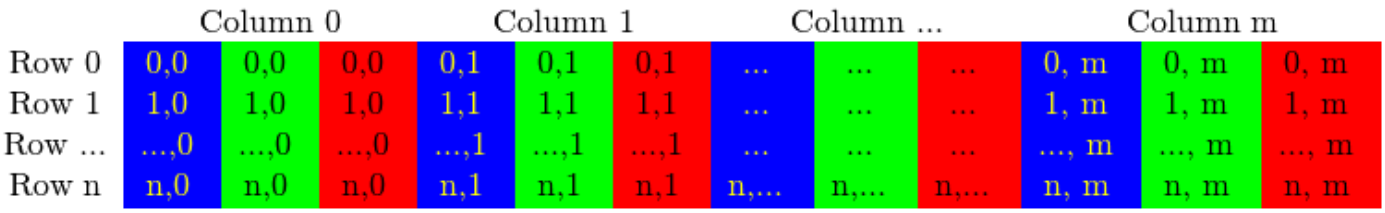
```
Rect rp(x, y, ширина, высота);//Задаём прямоугольник по которому будем выделять ОИ, где (x,y) - координаты его верхнего левого угла. При их изменении смещается весь прямоугольник.
Mat roi = img(rp);//Вот так просто теперь переменная roi указывает на область изображения img с габаритами и координатами из rp
```

!ВАЖНО: При попытке создать ОИ, которая выходит (даже частично) за пределы исходного изображения, функция выдаст сообщение об ошибке и прекратит выполнение программы.

Полезная информация!!!:

1. Существует более эффективный способ доступа к элементам матрицы (особенно при их последовательном переборе) при помощи ***указателей*** (или их модификаций - ***итераторов***), [http://docs.opencv.org/doc/tutorials/core/how_to_scan_images/how_to_scan_images.html]

Хранение пикселей строк и стобцов в объекте Mat



Пример доступа к данным (относительно безопасный)

```
int channels = I.channels();

    int nRows = I.rows;
    int nCols = I.cols * channels;//число столбцов с учетом цветовых каналов

    if (I.isContinuous())
    {
       nCols *= nRows;
       nRows = 1;
    }

    int i,j;
    uchar* p;
    for( i = 0; i < nRows; ++i)
    {
       p = I.ptr<uchar>(i);//указатель на строку
       for ( j = 0; j < nCols; ++j)
       {
          p[j] = table[p[j]]; // адресация в массиве строки
       }
    }
```

Не очень безопасный (через указатели)

```
uchar* p = I.data;

for( unsigned int i =0; i < ncol*nrows; ++i)
   *p++ = table[*p];
```

Безопасный

```
const int channels = I.channels();
  switch(channels)
  {
  case 1:
    {
      MatIterator_<uchar> it, end;
      for( it = I.begin<uchar>(), end = I.end<uchar>(); it != end; ++it)
        *it = table[*it]; //*it – получение объекта от итератора
      break;
    }
  case 3:
    {
      MatIterator_<Vec3b> it, end;
      for( it = I.begin<Vec3b>(), end = I.end<Vec3b>(); it != end; ++it)
      {
        (*it)[0] = table[(*it)[0]];//*it – получение объекта от итератора
        (*it)[1] = table[(*it)[1]];
        (*it)[2] = table[(*it)[2]];
      }
    }
  }
```

2. Чтобы полностью скопировать содержимое одной переменной ***Mat*** в другую следует использовать метод ***Mat::clone()***.

Mat img2 = img.clone();

А при использовании оператора **« = »** копируются все члены переменной ***Mat***, но при этом не копируется блок памяти под данные матрицы, на который указывает указатель ***Mat::data***. В итоге получаем тот же ОИ, но размером с исходное изображение.

2. Для оценки производительности в OpenCV до 3-ей версии был специальный класс ***TickMeter***, который выполнял функции таймера. Если использовать OpenCV 3.0.0, то необходимо добавить в проект следующий класс:

```
class CV_EXPORTS TickMeter
{
public:
  TickMeter();
  void start();
  void stop();
```

```
    int64 getTimeTicks() const;
    double getTimeMicro() const;
    double getTimeMilli() const;
    double getTimeSec()   const;
    int64 getCounter() const;

    void reset();
private:
    int64 counter;
    int64 sumTime;
    int64 startTime;
};

std::ostream& operator << (std::ostream& out, const TickMeter& tm);


TickMeter::TickMeter() { reset(); }
int64 TickMeter::getTimeTicks() const { return sumTime; }
double TickMeter::getTimeMicro() const { return  getTimeMilli()*1e3; }
double TickMeter::getTimeMilli() const { return getTimeSec()*1e3; }
double TickMeter::getTimeSec() const { return (double)getTimeTicks() / cv::getTickFrequency(); }
int64 TickMeter::getCounter() const { return counter; }
void TickMeter::reset() { startTime = 0; sumTime = 0; counter = 0; }

void TickMeter::start(){ startTime = cv::getTickCount(); }
void TickMeter::stop()
{
    int64 time = cv::getTickCount();
    if (startTime == 0)
            return;
    ++counter;
    sumTime += (time - startTime);
    startTime = 0;
}

std::ostream& operator << (std::ostream& out, const TickMeter& tm) { return out << tm.getTimeSec()
<< "sec"; }
```

ЗАДАНИЕ

1. Создать отдельную функцию, которая бы сглаживала одноканальное изображение при помощи фильтра по среднему значению пикселей внутри ядра размером 3x3 (в лекциях обозначен как box filter). Для этого рекомендуется использовать ОИ. Функция вычисления, однако, должна иметь возможность менять размер окна
2. Проверить правильную работу созданной функции при помощи функции из OpenCV void blur(InputArray **src**, OutputArray **dst**, Size **ksize**). Для этого необходимо программно сравнить изображения, проверяя равенство каждого пикселя одного изображения другому пикселю. Для этого рекомендуется использовать способ ***The Core Function*** из ранее приведённой ссылки. Вывести значение их схожести в виде процентов. Если схожесть 98%-100%, то твоя функция работает исправно. Отличие в немного % может объясняться тем, что вы не обработали рамку по краям изображения. Иногда эту рамку оставляют (как в нашем простом случае), иногда обрабатывают, достраивая дополнительные пиксели или беря их с других участков изображения. Необходимо вывести оба изображения.
3. Измерить время работы каждой функции и вывести его в консоль. Вывести результат измерений.

Примерно, как должно выглядеть:

Изображение ***img_dif*** выведено для отображения разницы между ***img_s2*** и ***img_s3***.

4. Отфильтровать изображение функцией Гаусса и сравнить результат с box filter. Субъективно оценить качество изображения. Вывести оба изображения и сравнить их путем наложения и вывода разности двух фильтраций. Улучшить отображение выведенной разницы применив логарифмическое преобразование изображений.
5. Реализовать unsharp mask с фильтром Гаусса и Box фильтром. Оба фильтра необходимо применить с одинаковым коэффициентом резкости (см. выше). Сравнить оба изображения путем наложения и вывода разностного изображения.
6. Реализовать фильтр Лапласа (руками, свой). Вывести результат.
7. Реализовать unsharp mask с фильтром Лапласа. Вывести результат и сравнить с п.5.