# Report exercise 2 - Convolution filter

In this exercise you have to program an OpenCV c++ function:

`void linearFilter(cv::Mat input, cv::Mat kernel, cv::Mat &output)`

for performing convolution on an image. The function specifications are as follows:

- The input image is an M × N single channel 8-bit unsigned integer image (grayscale).
- The kernel is a single channel 32-bit floating point image. Its size is 3×3 and it has the anchor point in the center of the kernel.
- The output is the convolution $I * k$ between the input image $I$ and the kernel $k$.
- The output image is an M × N single channel 8-bit unsigned integer image (similar to the input image).
- You have to use zero padding to keep the image dimensions unchanged.

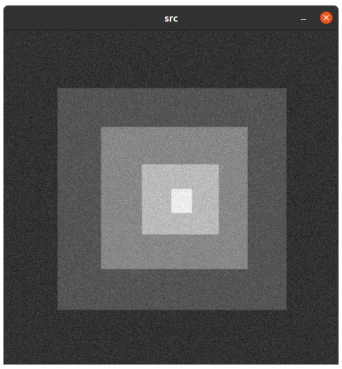
To test your method you can use the following main.cpp as template. It first applies the convolution operation on an image (the path is provided as an argument). It then applies the filter to an impulse function.

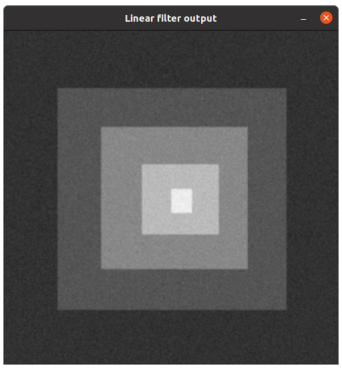
### main.cpp

```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <iostream>

int main(int argc, char* argv[]){
    //Load image as grayscale
    if(argc != 2){
        std::cout << "Usage: ./main <imagefile.jpg/png>"<< std::endl;
        return -1;
    }
    std::string filename = argv[1];
    cv::Mat src = cv::imread(filename, cv::IMREAD_GRAYSCALE);
    cv::namedWindow("src");
    cv::imshow("src",src);
    cv::waitKey(0);
    CV_Assert(src.type() == CV_8UC1);

    //Create uniform 3x3 kernel
    cv::Mat kernel(3,3, CV_32FC1, cv::Scalar(1.0/9.0));
    CV_Assert(kernel.type() == CV_32FC1);
```

```
    //Apply linear filter
    cv::Mat output;
    linearFilter(src, kernel, output);
    cv::namedWindow("Linear filter output");
    cv::imshow("Linear filter output",output);
    cv::waitKey(0);

    //Test with custom kernel on impulse image
    cv::Mat impulse = cv::Mat::zeros(5,5,CV_8UC1);
    impulse.at<uchar>(2,2) = 1;

    cv::Mat customKernel = (cv::Mat_<float>(3,3) << 1,2,3,4,5,6,7,8,9);
    linearFilter(impulse,customKernel,output);

    for(int i=0; i<output.rows; i++){
        for(int j=0; j<output.cols; j++){
            std::cout << (int) output.at<uchar>(i,j) << " ";
        }
        std::cout << std::endl;
    }
    return 0;
}
```

## Input/Output

**input image:** 


**output image:** 


**Output of custom filter applied to impulse**

```
0 0 0 0 0
0 1 2 3 0
0 4 5 6 0
0 7 8 9 0
0 0 0 0 0
```