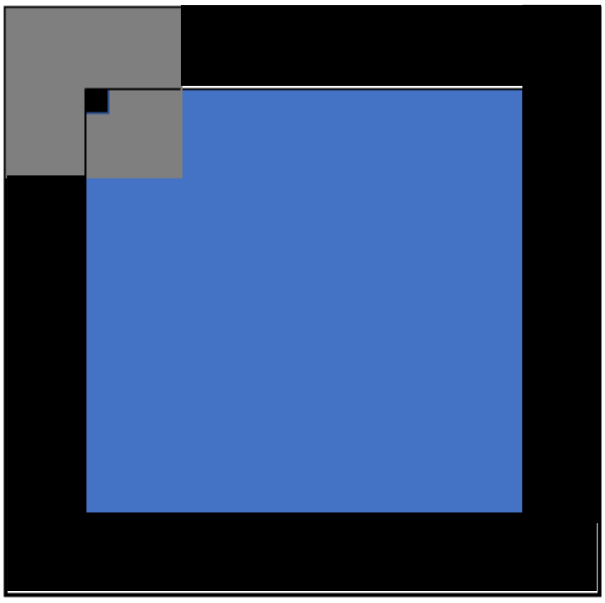
COMP408 Assignment 1, Prepared by Arda Kırkağaç, 49799

In this project, our aim was to create a 2D filter which can both process grayscale and color images. It turned out that you should specify the properties and relevant conversions (such as CV_8UC1 for single-channel grayscale image, and CV_64FC3 for 3-channel colored image) in the main() function. In the end, I managed to create a function that outputs a grayscale image if the inputs are grayscale, and colored if the inputs are colored. However; as discussed in the lecture on Thursday, a working filter only on colored images will be considered as sufficient. Hence my final code only includes the algorithm for colored images.

In the filter algorithm, I first learned the dimensions of the filter and the images, and created empty matrices to fill them appropriately, as C++ requires proper initializations. The dimensions will be also used in the "for" loops for the filtering. I also checked for filter size to be odd*odd.

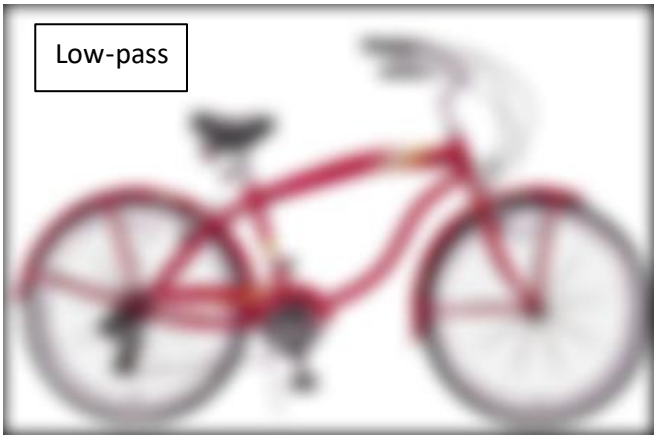
```
//check for filter dimensions, learn image and filter dimensions
int filterRows = filter.rows;
int filterCols = filter.cols;
if (filterRows % 2 != 1 || filterCols % 2 != 1) {
    return im;
}
int imageRows = im.rows;
int imageCols = im.cols;
int channelNo = im.channels();
//////////////////////////////////////////////////////////////

//defining matrices/////////////////////////////////////////////////////////////////////////////
Mat outI;
Mat splitted[3];
Mat filtered[3] = { Mat(imageRows,imageCols,CV_64F),
    Mat(imageRows,imageCols,CV_64F),
    Mat(imageRows,imageCols,CV_64F)
};
//////////////////////////////////////////////////////////////////////////////////////////////
```

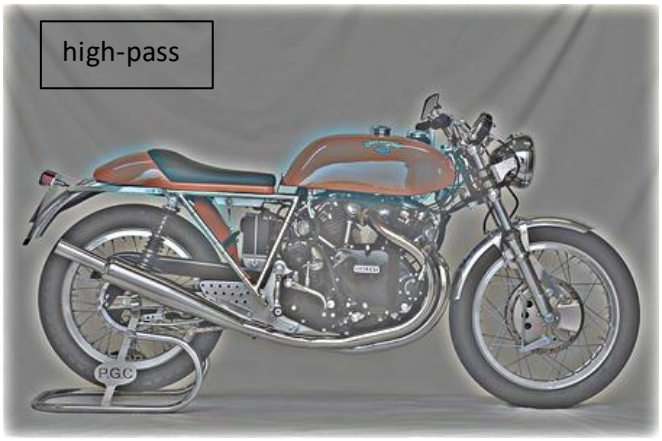


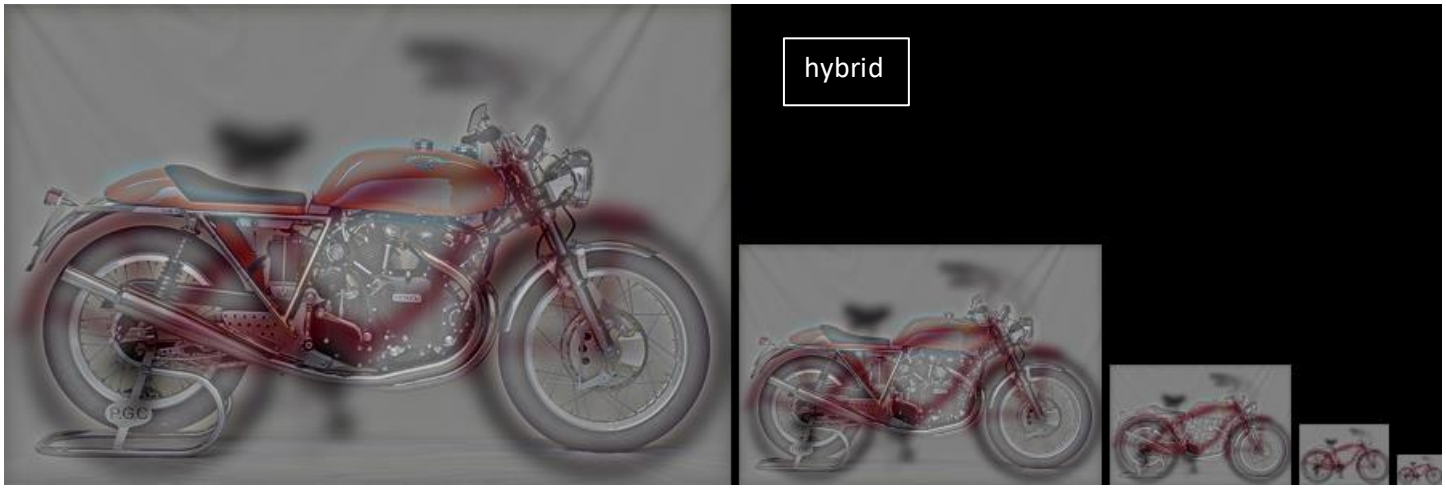
This is a rough sketch which tries to explain cross-correlation (filtering) with zero padding. Additional zeros (black pixels, looking as a frame) are added to support the operation for the edge and corner pixels. A zero padding of (filter_size/2)-1 would be enough for our operations. Relevant code is below, where I also specify the padded values as zeros:

```
copyMakeBorder(im, im, (filterRows - 1) / 2, (filterRows - 1) / 2, (filterCols - 1) / 2, (filterCols - 1) / 2, BORDER_CONSTANT, 0);
```

I made additional assignments to matrices to initiate filtering. For example, “split” function is used to separate channels of the padded image, since we need to apply filtering to each channel separately. There is one main cross-correlation algorithm, properly working for colored images. The matrix which will be used for the output is at the same size as the input image already. What I did was to perform correlation in the padded image and to print it into another matrix. I also used print operations to keep track of the operation at the command window. You can see my source code attached with this submission. My example run for two colored images is visualized below:







Here you saw a hybrid image of a bicycle and a motorcycle, in which the bicycle covered low frequencies, whereas the motorcycle covered high ones. I chose to use a kernel of 25x25, where sigma=6. The black framing in the low-pass image, and the white framing in the high-pass image is mainly caused by zero padding. We used zero padding for achieving a convenient filtering operation.

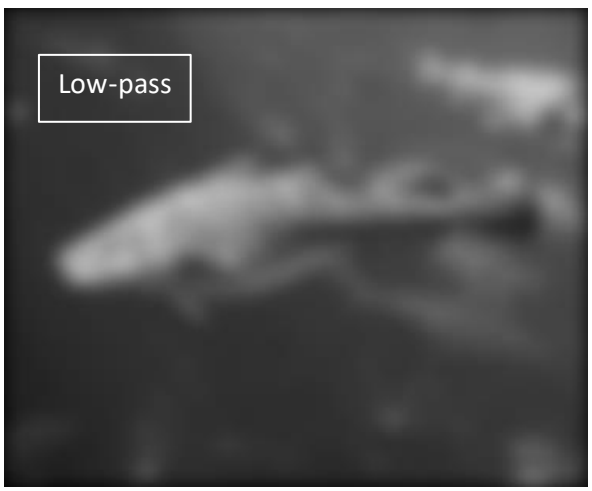
To do processing with grayscale images, one can use this line:

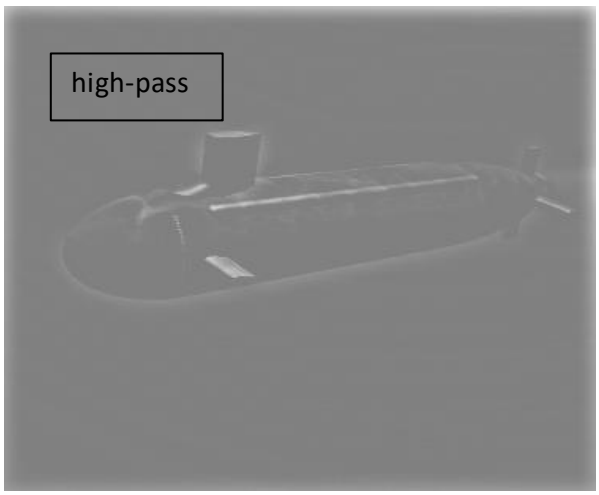
```
Mat image1 = imread("../data/bicycle.bmp",0);
```

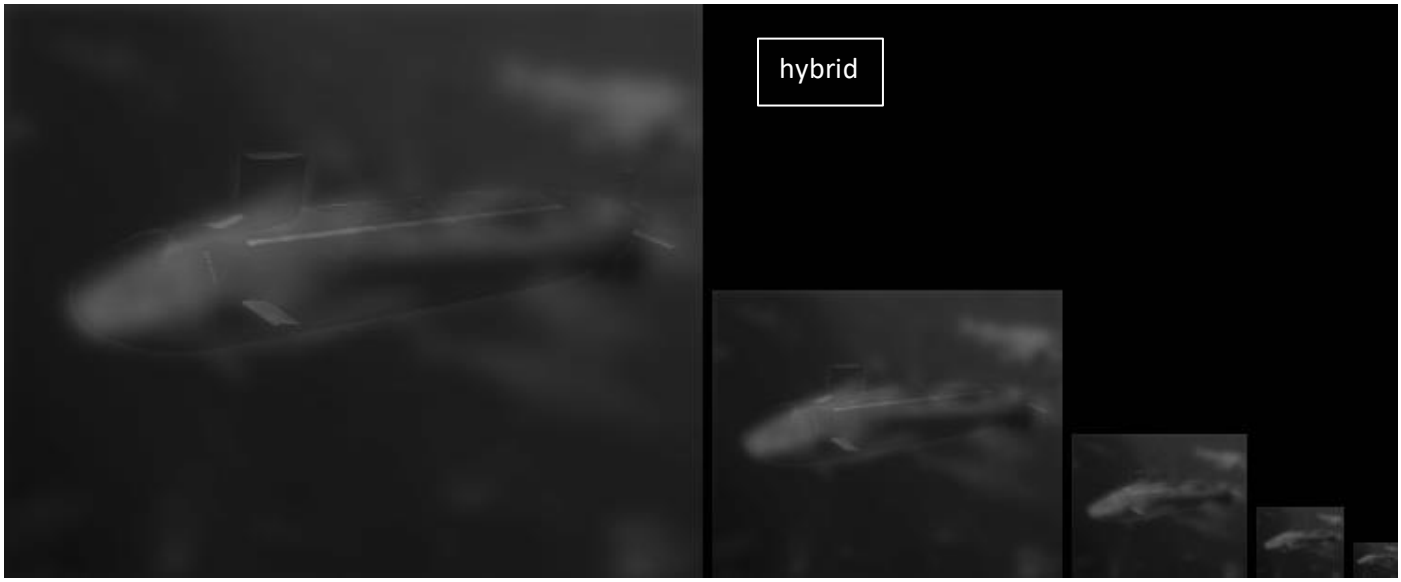
The zero at the end of the argument will cause the compiler to internally read the image as a grayscale one. Furthermore, we also need to change each CV_8UC3 with CV_8UC1 (the last digits indicate the number of channels), such as in concatenate function, or visualization lines. Examples include:

```
hconcat(output, Mat::ones(original_height, padding, CV_8UC1), output);
low_frequencies.convertTo(low_frequencies, CV_8UC1);
```

Although my final code does not host the algorithms for grayscale images, my earlier example run for two grayscale images is visualized below (sigma=7, kernel size=29):



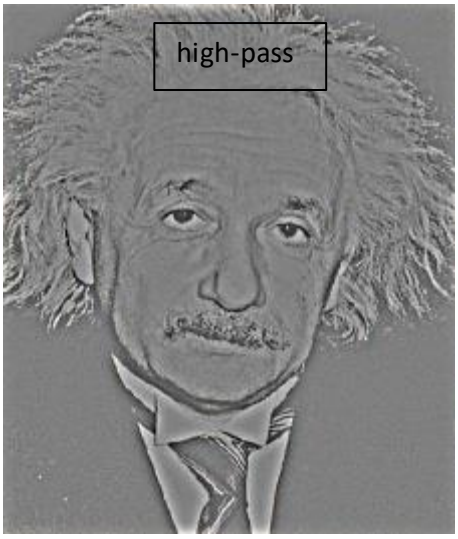


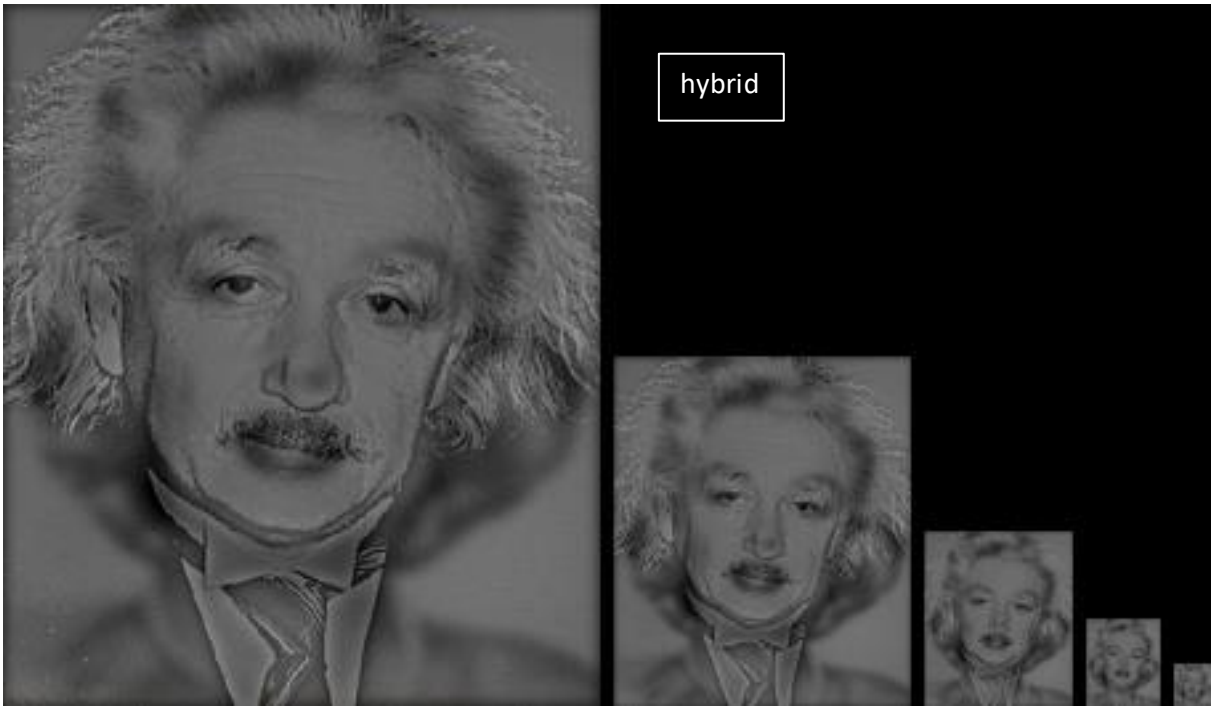


Here you saw a hybrid image combination of a submarine and a fish. When you look the largest image, the submarine image is obvious, but as the image is being shrunk, one can start to see the fish and ignore high frequencies.

Below is an additional hybrid image of Albert Einstein and Marilyn Monroe. Although those images look like grayscale, they have three channels (sigma=3). Other images and their processed outputs will be provided in the submission.s


low-pass


high-pass


hybrid