Image enhancement using blur mask in the spatial domain

Exercise goals:

- Learn how to enhance an image by applying an unsharp mask
- Learn how to apply a convolution to the image.
- Learn how to use the openCV GUI using controls with slide bar.

Minimum requirements:

We are creating a program that will do the following operation to the image
$O = (g+1) \cdot I - g \cdot I_L$

Where is the original image I, I_L is a low pass version of the image, and g is a factor controlling the gain of the operation.

- The input image is grayscale (force grayscale image when loading)
- Implement thw two following functions:

```
//Creates a Box filter of radius r>0.
cv::Mat createBoxFilter(int r);

//computes the convolution of an image and a filter. The output
//image must have the same dimenions as the input image. Fill with
//zeros the borders of the image not processed.
//Preconditions: in.type()==CV_32FC1 && filter.type()==CV_32FC1.

void convolve(const cv::Mat& in, const cv::Mat& filter, cv::Mat&
out);
```

- The program must have the following command line behaviour:

```
unsharp [-r <float>] [-g <float>] [-f int] <input img> <output
img>
```

- The parameter ‘r’ controls the size of the filter 2r+1,. Please consider that ‘r’ is an integer value with default value 1.
- The parameter ‘g’ is a floating value in the range [0,10] with default value 1, controlling the gain of the filter.

Optional requirements:

- Implement the option [-f int] which allows to employ a Gaussian filter instead of the Box filter. If 'f': 0->box, 1->gaussian filter. To do so, create the function

```
//Creates a  Gaussian filter of radius r>0.
cv::Mat createGaussianFilter(int r);
```

- Add the flag *circular* to the previous function convolve. When activated, the border pixels will be processed using the opposite image pixels.

```
void convolve(const cv::Mat& in, const cv::Mat& filter, cv::Mat&
out, bool circular=false);
```

- Add support for color images working in the HSV color space, applying the filter only in the V channel and then moving back to the RGB color space for storing the result.
- Add the interactive flag [-i] . In this mode, the original and resulting images are shown to the user in windows. Additionally, you must add sliders to control the parameters 'r' and 'g' and recompute the resulting image based on them. The application must finish when the user press ESC, and save the image to a file if the user press ENTER.

Resources:

For creating slider for the window use: cv::createTrackBar.

Image example 1 (monocroma).
Image example 2 (color), Mask for image 2.