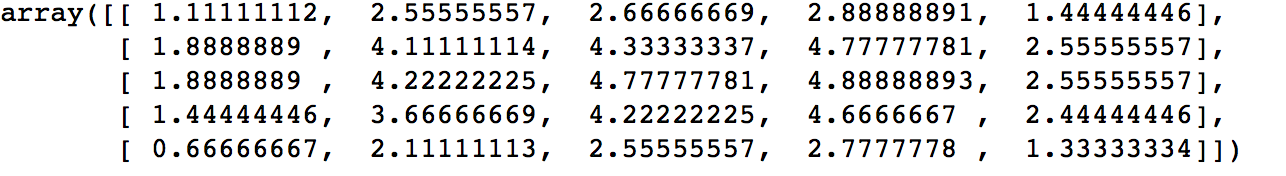
**Computer Vision – Assignment 1**

*Ashwin Ramanathan*

**Q1 Image filtering and enhancement**

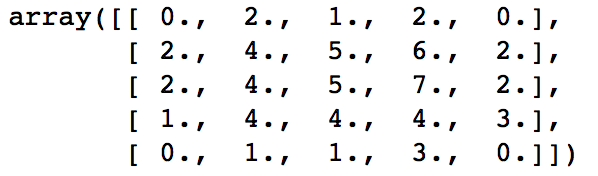
Code: solution-part1.ipynb

1. The convolution of the given matrix with a mean filter is



The convolution was performed by padding the matrix with zero around the edges. As it is a mean filter, it can be seen that the high values , especially at the edges, have decreased ,and points with low values have a higher value after convolution.

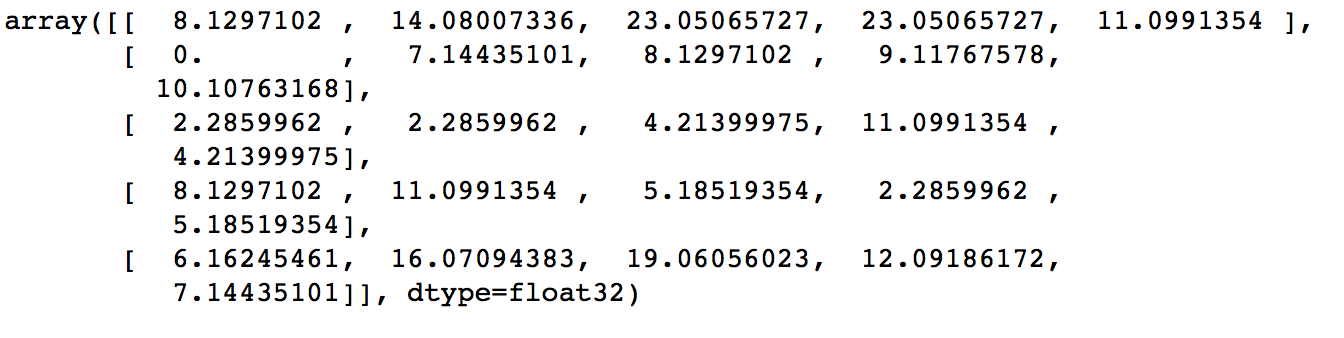
1. Convolution with median filter



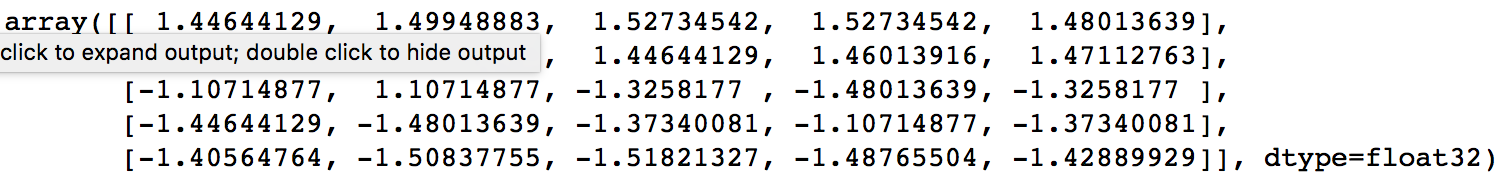
Convolution with a median filter replaces a point’s values with the median of values in the kernel window, whereas a mean filter replaces a point’s value with the avg of all values in the kernel window.

1. After applying the Sobel operator

Gradient magnitude:



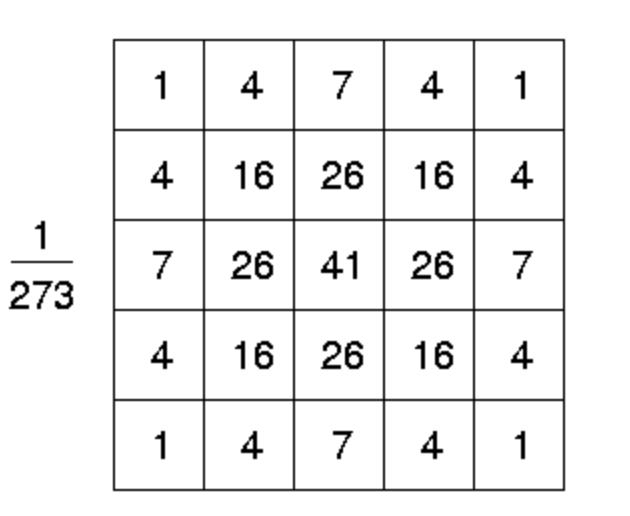
Gradient direction:



Each point in the gradient magnitude matrix provides the magnitude of the gradient of the corresponding pixel point. Gradient magnitude is give by *Gmag = np.sqrt(Gx\*Gx + Gy\*Gy)* where Gx and Gy are partial derivative equivalents (w.r.t x and y).

Gradient direction is given by *GradDir = np.arctan(Gy,Gx)*

1. a. A Gaussian filter is a very good example of a distance based filter. The Gaussian kernel has peak values at the center and the values decrease gradually based on the kernel’s sigma value. This can be seen in the below image.



Hence when a convolution is performed, then values near to the image block’s center (a *k x k* block with which convolution is performed in an iteration) are given higher weights as compared to pixels which are far away (like the corners), making this a distance based filter.

b. A range filter is an example of a filter which considers only the distance between pixel values. It replaces the pixel with the difference between maximum and minimum pixel values in its local neighborhood.

c. A bilateral filter is a good example of a filter which considers both distance as well similarity (or in a way, distance) between pixel values. Instead of taking a weighted average of pixels (like in the case of Gaussian) it considers a function of pixel value as well. Due to this, pixels which do not have similar values (for example, pixels which are on opposite sides of an edge) will have minimal impact on the pixel after convolution. This filter can be used to perform blurring while preserving edges at the same time. (Implementation used: skimage.restoration.denoise\_bilateral)

|  |  |
| --- | --- |
|  |  |

|  |
| --- |
|  |

The images in the previous page display the effect of applying the filters to the three images. In all the three sets of images, it can be seen that

1. Gaussian and Bilateral filter are successful in decreasing noise to quite an extent. Although, images after bilateral filtering have sharper edges than Gaussian filtered images.
2. Range filter works like an edge detector, and the edge identification capabilities of this filter is not impacted by noise. However, the range filter (and by extension, any pixel value based filter) is not very capable in removing noise from an image.

|  |  |
| --- | --- |
|  |  |
|  | |

1. For this part, the task was to apply unsharp masking on the image. Unsharp masking is a process which is often used to slightly sharpen images. The process:
2. The image is blurred (usually using a Gaussian filter) and subtracted from the original image. This leaves behind those parts of the image which have comparatively higher intensity values, often, near of around the edges.
3. The result of the previous step is added to the image again (often with a multiplication factor). This amplifies the regions with high intensity, and makes the image look sharper.

The effect of unsharp masking can be seen in the image. The images with unsharp masking seem to have higher contrast and sharper edges.

**Q2. Color quantization with k-means**

Code: solution-part2.ipynb

1. Function name: quantize(img,k)

This function takes the image as an input, converts it into an array of dimension rows\*cols,3 and then performs kmeans clustering with k clusters. The result of the prediction is the cluster to which each point is assigned, so a new array is created which replaces the value of each pixel (r,g &b) with the value of the centroid of the cluster to which it belongs. This array is converted to the shape of the input image and returned.

1. Function name: labQuantizeAndCombine(img,k)

The rgb image is converted to lab using skimage.rgb2lab. The image was converted to 0-1 scale before conversion.

1. Function name: calculateSSD(im1, im2)

This function calculates the SSD by summing the square of differences in pixel values (r,g &b) of image 1 and image2. Since both images are expected to be of the same size (this function will be used to calculate SSD between the original and quantized image), conditions of unequal sizes haven’t been handled.

1. Function name: plotHists(img,k)

This function first performs quantization of the l channel and then plots the histograms.

1. Function name: compileResults(img, k)

**Part 3: Edge Detection**

A.1

The underlying principle in finding edges is that the intensity varies at the edges. This variation is successfully captured by the partial derivatives (w.r.t x and y axes). All the three edge detection methods mentioned below use partial derivatives.

Sobel filtering:

Gaussian Laplace filtering: The first step in Laplace of Gaussian edge detection is filtering the input image with the Gaussian. Then, the Laplacian based edge detection points of an image can be detected by finding the zero crossings of the second derivative (of the image intensity), which gives the edges.

Canny Edge Detection: