**from** **PIL** **import** Image

**import** **numpy** **as** **np**

**import** **math**

**from** **scipy** **import** signal

**def** boxfilter( n ):

*#Check if its even, throw an error if it is*

*#Box filter should be odd, to have better centre precision*

**if** (n % 2 ==0):

**return** **AssertionError**('Dimension must be odd')

**else**:

*#All values in array should sum up to equal 1*

*#Thats why we divide by 1/n^2 to get the number*

arrayNumber = 1/math.pow(n, 2)

**return** np.full((n, n), arrayNumber)

**def** gauss1d(sigma):

*#As per instructions, length should be 6 times sigma and rounded up*

arrayLength = math.ceil(sigma\*6)

*#in case it's an even number add one to make it odd*

**if** arrayLength % 2 == 0:

arrayLength = arrayLength+1

*#Want to generate a 1D array, centred at 0, but going negative*

*#and positive in each relative direction.*

*#For example: sigma 1.0 will have sigma\*6: [-3, -2, -1, 0, 1, 2, 3]*

*#Hence it goes up to half the array length (3) for each side of 0*

gaussArray = np.arange(-(math.floor(arrayLength/2)),

math.ceil(arrayLength/2))

*#For each number in array, we want to apply the gaussian function*

*#dependant on their length from the middle*

result = map(**lambda** x: round(math.exp(-((x\*\*2)/(2\*(sigma\*\*2)))), 8), gaussArray)

*#Normalize each value with the constant of 1/(sum of the array)*

normalizedResult = np.array(result)\*(1/(np.sum(result)))

**return** normalizedResult

**def** gauss2d(sigma):

*#Find the 1D gaussian given sigma*

gauss1Array = gauss1d(sigma)

*#Make it 2D by adding an axis, so we can transpose it*

gauss1Array = gauss1Array[np.newaxis]

*#Transpose the 1D (now 2D) array*

gauss1Transpose = np.transpose(gauss1Array)

*#Convolve the 2D array with its transpose*

**return** signal.convolve2d(gauss1Array, gauss1Transpose)

**def** gaussconvolve2d(array, sigma):

*#Get the convolution given sigma*

ifilter = gauss2d(sigma)

*#Apply the filter (convolution) to image (array)*

**return** signal.convolve2d(array, ifilter, 'same')

**import** **Hmk2**

**from** **PIL** **import** Image

**import** **numpy** **as** **np**

**print**("1) BOXFILTER TESTS: **\n**")

**print**(Hmk2.boxfilter(3))

*"""*

*[[ 0.11111111 0.11111111 0.11111111]*

*[ 0.11111111 0.11111111 0.11111111]*

*[ 0.11111111 0.11111111 0.11111111]]*

*"""*

**print**(Hmk2.boxfilter(4))

*"""*

*AssertionError: Dimension must be odd*

*"""*

**print**(Hmk2.boxfilter(5))

*"""*

*[[ 0.04 0.04 0.04 0.04 0.04]*

*[ 0.04 0.04 0.04 0.04 0.04]*

*[ 0.04 0.04 0.04 0.04 0.04]*

*[ 0.04 0.04 0.04 0.04 0.04]*

*[ 0.04 0.04 0.04 0.04 0.04]]*

*"""*

**print**("**\n\n\n**2) GAUSS1D TESTS: **\n**")

**print**(Hmk2.gauss1d(0.3))

*"""*

*[ 0.00383626 0.99232748 0.00383626]*

*"""*

**print**(Hmk2.gauss1d(0.5))

*"""*

*[ 0.10650698 0.78698605 0.10650698]*

*"""*

**print**(Hmk2.gauss1d(1))

*"""*

*[ 0.00443305 0.05400558 0.24203623 0.39905028 0.24203623 0.05400558*

*0.00443305]*

*"""*

**print**(Hmk2.gauss1d(2))

*"""*

*[ 0.0022182 0.00877313 0.02702316 0.06482519 0.12110939 0.17621312*

*0.19967563 0.17621312 0.12110939 0.06482519 0.02702316 0.00877313*

*0.0022182 ]*

*"""*

**print**("**\n\n\n**3) GAUSS2D TESTS: **\n**")

**print**(Hmk2.gauss2d(0.5))

*"""*

*[[ 0.01134374 0.0838195 0.01134374]*

*[ 0.0838195 0.61934704 0.0838195 ]*

*[ 0.01134374 0.0838195 0.01134374]]*

*"""*

**print**(Hmk2.gauss2d(1.0))

*"""*

*[[ 1.96519284e-05 2.39409418e-04 1.07295860e-03 1.76900966e-03*

*1.07295860e-03 2.39409418e-04 1.96519284e-05]*

*[ 2.39409418e-04 2.91660281e-03 1.30713073e-02 2.15509423e-02*

*1.30713073e-02 2.91660281e-03 2.39409418e-04]*

*[ 1.07295860e-03 1.30713073e-02 5.85815363e-02 9.65846250e-02*

*5.85815363e-02 1.30713073e-02 1.07295860e-03]*

*[ 1.76900966e-03 2.15509423e-02 9.65846250e-02 1.59241126e-01*

*9.65846250e-02 2.15509423e-02 1.76900966e-03]*

*[ 1.07295860e-03 1.30713073e-02 5.85815363e-02 9.65846250e-02*

*5.85815363e-02 1.30713073e-02 1.07295860e-03]*

*[ 2.39409418e-04 2.91660281e-03 1.30713073e-02 2.15509423e-02*

*1.30713073e-02 2.91660281e-03 2.39409418e-04]*

*[ 1.96519284e-05 2.39409418e-04 1.07295860e-03 1.76900966e-03*

*1.07295860e-03 2.39409418e-04 1.96519284e-05]]*

*"""*

**print**("**\n\n\n**4) GAUSS2D TESTS: **\n**")

im = Image.open('van.png')

*#Open original image*

im.show()

*#Convert to grayscale*

im = im.convert('L')

im\_array = np.asarray(im)

*#Apply the filter using the gaussconvolve2d function we coded*

imconvolved = Hmk2.gaussconvolve2d(im\_array, 3)

*#Convert the type or PIL throws as an error*

imconvolvedImage = Image.fromarray(imconvolved.astype(np.uint8))

*#Save the filtered image*

imconvolvedImage.save('van-filtered.png', 'PNG')

*#Open filtered image*

imconvolvedImage.show()