Computer Vision 1 - Assignment 2
Linear Filters: Gaussians and Derivatives

Selene Baez Santamaria (10985417) - Andrea Jemmett (11162929)

February 24, 2016

1 1D Gaussian Filter

Our function takes the same arguments as the built-in fspecial, but it generates different output. While our function generates a vector of size $kernelLength$, the *fspecial* function returns a rectangular or squared matrix dependant on the $hsize$ parameter.

2 Convolving an image with a 2D Gaussian

We use a value of $\sigma_x = 2$ and $\sigma_y = 2$ and a kernel length of 11 (as requested). First we call the *gaussian* function defined in the previous section to get two 1D filters, one with each σ. We multiply those 1D vectors to obtain a 2D gaussian kernel. To convolve the images with kernels we use the built-it function $conv2$. After testing the different options we found out the following:

full performs the full convolution. Visually, we note that the produced images have black frames on the edges since the image size is slightly (10 pixels) larger than the original image; this is due to the fact that the convolution operation "spreads" pixel intensities over neighboring pixels overflooding outside the original image size;

same returns the central part of the convolution, that is the image returned as output is of the same size of the input matrix A;

valid performs convolution but omits those parts of the convolution that are computed not exclusively using image pixels (image edges); the resulting image is indeed smaller than the original.

In order to check that both functions produce the same images we make a pixel per pixel numeric comparison. We check that the absolute value of the difference among the values is less that a small value $\epsilon = e^{-12}$. Hereby we show the original and filtered images using the *same* option:

Figure 1: Comparison for flowers.jpg. From left to right: original image, image filtered with 2D kernel and image filtered with 1D kernel. All images have a σ value of 2

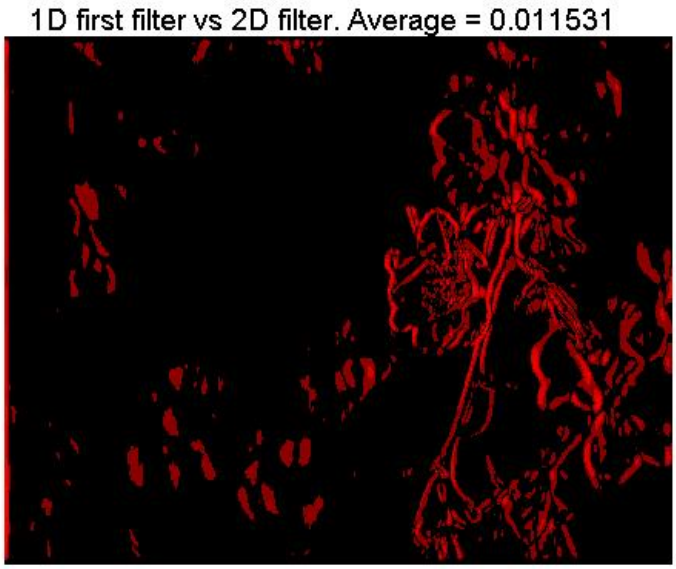

Figure 2: Heatmap showing the absolute difference per pixel between Step 1 of 1D filtering and 2D final filtered image for flowers.jpg. Reported mean error per pixel is 0.0115

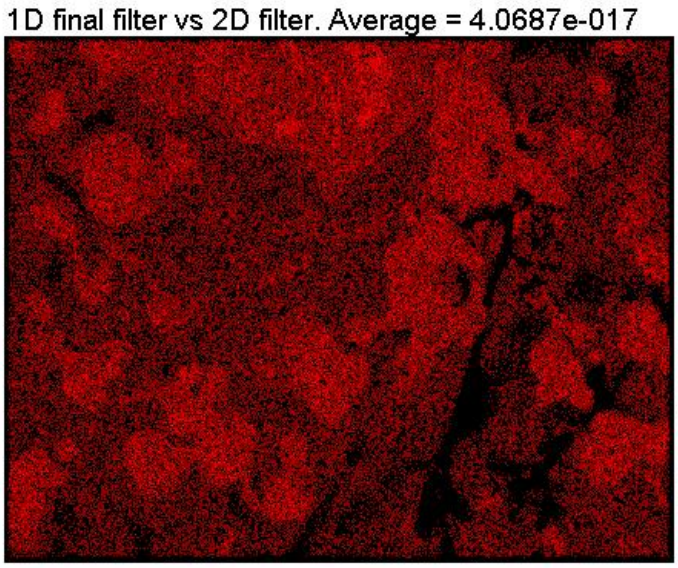

Figure 3: Heatmap showing the absolute difference per pixel between both final filtered images for flowers.jpg. Reported mean error per pixel is $4.0687e^{-17}$

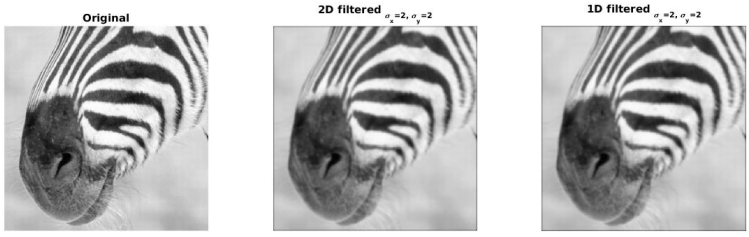

Figure 4: Comparison for zebra.png (original, 2D kernel, 1D kernel with $\sigma = 2$). Reported mean error per pixel is $5.5609e^{-17}$

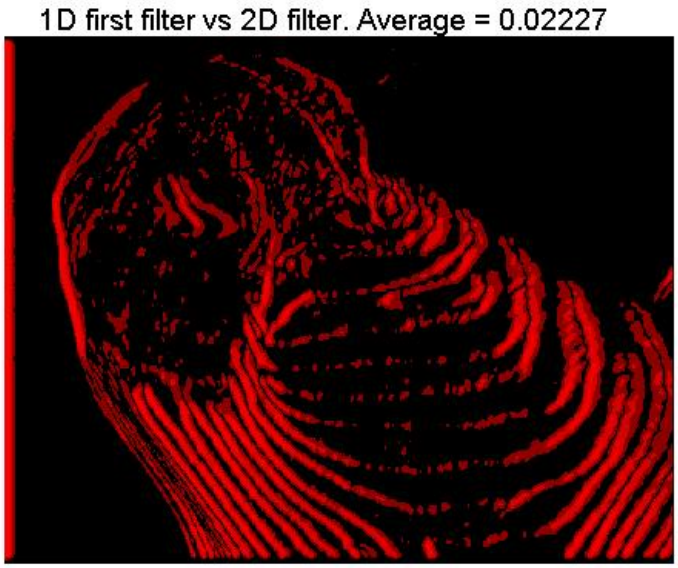

Figure 5: Heatmap showing the absolute difference per pixel between Step 1 of 1D filtering and 2D final filtered image for zebra.png. Reported mean error per pixel is 0.0223

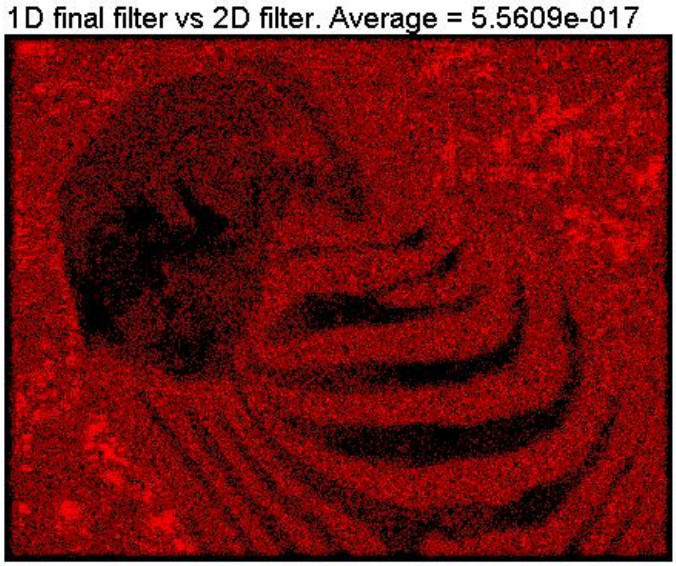

Figure 6: Heatmap showing the absolute difference per pixel between both final filtered images for zebra.png. Reported mean error per pixel is $5.5609e^{-17}$

Having kernel separability is helpful since the number of operations needed to perform the convolution operation are fewer than with a full 2D kernel. For example filtering an M-by-N image with a P-by-Q kernel would require $MNPQ$ operations. If the kernel is separable, we can apply the convolution in an ordered fashion, first by applying a vector kernel to columns and then filtering the results using a vector kernel on the rows. Filtering in these two steps requires respectively MNP and MNQ operations, so the total number of operations is $MN(P+Q)$, which is less than $MNPQ$.

3 Gaussian Derivative

We implemented a function called *gaussianDer* which computes the first order derivative of a gaussian kernel. The function takes as parameters a path to the image to filter, a gaussian kernel and a σ value. We try this filter with the same image with different sigma values and a kernel length of 11. The resulting filtered images are shown in Figure 7. The Gaussian derivative may be used to find edges in am image. An application could

be countoruing an image. From the Figure it is possible to note that for values of the standard deviation smaller than 1 and bigger than 2 the filter is almost unable to detect any edge, achieving the best results with $\sigma = 1.5$.

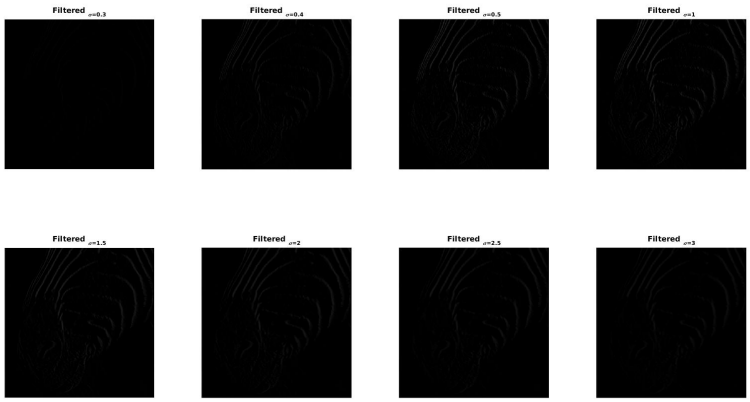

Figure 7: Application of the first order Gaussian derivative to the Zebra image with different σ values (from top left: $0.1, 0.2, 0.3, 0.4, 0.5, 1, 1.5, 2, 2.5, 3$).

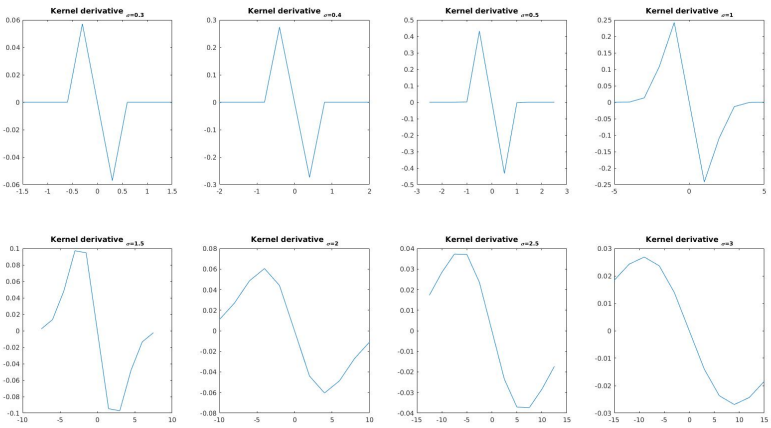

Figure 8: First order Gaussian derivative plotted on a range in $[-5\sigma, 5\sigma]$

From Fig. 8 is easily noticeable how the shape of the kernel signal affects the filtering results. For smaller sigma values the first order Gaussian

derivative is sharper and for values smaller than 1 it is unable to detect edges small enough. Vice versa happens with bigger values of σ for which the filter is unable to detect enough big edges.