2018

Digital Image Processing

Georgia Petridou

7964

5/12/2018

Contents

Import

In this paper, before explaining each section of queries, an introduction to the preprocessing before executing the queries will be provided.

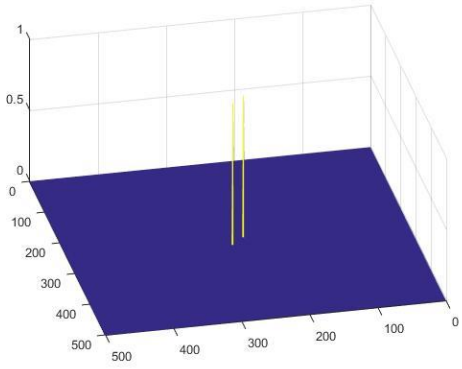
If in the Fourier transform of an image two symmetrical impulses δ(ω-ω0) are observed with respect to the center of the image, then this means that the image in the spatial field will display a spatial sine which will have a direction proportional to the direction of ω0 and a spatial frequency proportional to the magnitude of ω0.

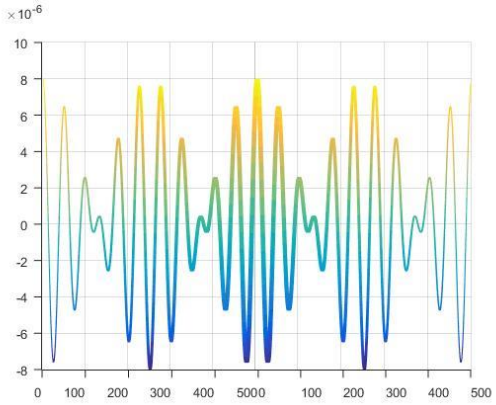
For this reason, when, with the suggestion of the pronunciation, such a function F is tested in MATLAB and visualized three-dimensionally in the frequency domain, as well as in the spatial domain, it gives us the following results.

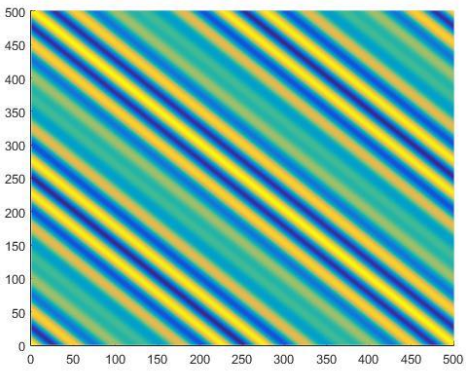
$$\boldsymbol{F(u1,u2)} = \begin{cases} 1+0j \ , for\ u1 = m\ and\ u2 = l \\ 1+0j \ , for\ u1 = M-m\ and\ u2 = M-l \\ 0 \ , elsewhere \end{cases}$$

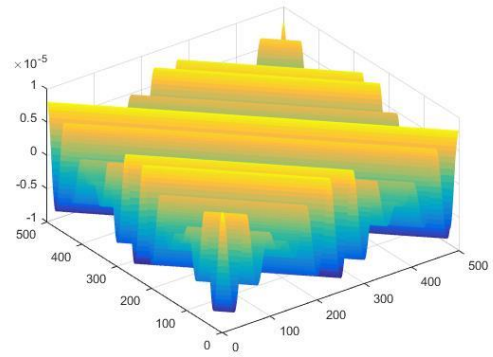
For m = 260 and l = 260 (M=500) we have the three-dimensional diagrams:

(the first corresponds to F(u1,u2) and the next three to f (n1,n2) (different perspective each time))









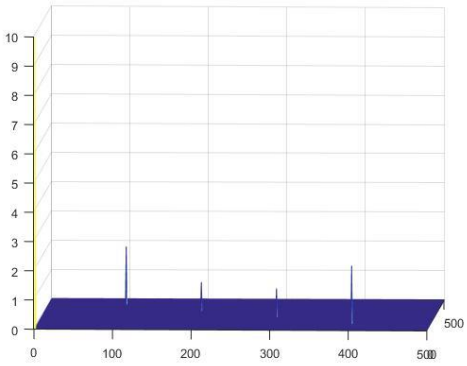
If a dc component is now placed in the function F and another pair of symmetrical impulses is added to it, then as we will observe, the center of oscillation will rise above zero

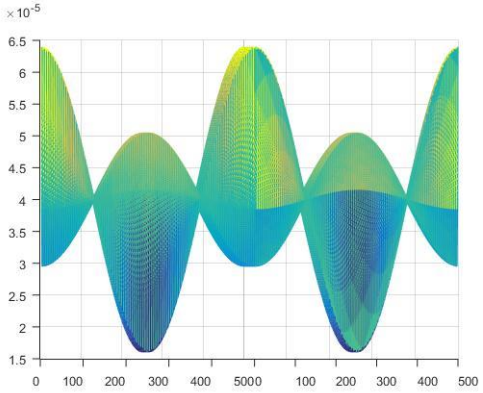
and the ripple will be denser due to the superposition of the sinusoids caused by the individual pairs of impulses in the frequency.

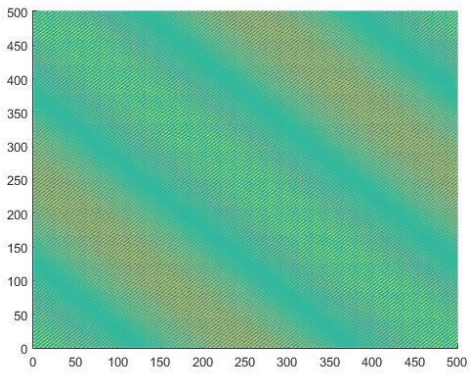
So, for

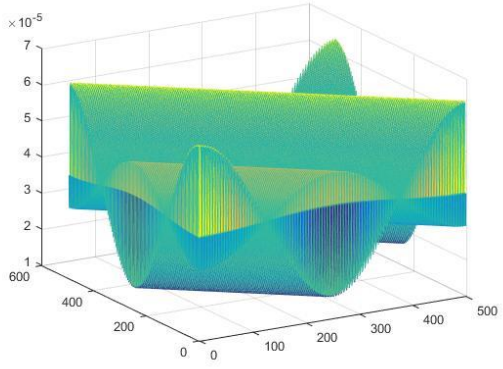
$$\boldsymbol{F(u1,u2)} = \begin{cases} 1+0j \ , for\ u1 = m\ and\ u2 = l \\ 1+0j \ , for\ u1 = M-m\ and\ u2 = M-l \\ 2+0j \ , for\ u1 = n\ and\ u2 = k \\ 2+0j \ , for\ u1 = M-n\ and\ u2 = M-k \\ N \ , for\ u1 = 1\ and\ u2 = 1 \\ 0 \ , elsewhere \end{cases}$$

With m = 200, l = 300, n = 400, k = 100 and N = 10 (M=500) we have:









Section 1

Script Demo1

The given image is loaded and the function implemented for its filtering is called.

Filtering in the frequency domain:

To filter an image in the frequency domain, the function myFiltFreq.m was implemented, which initially stores the dimensions of the image it accepts as an argument as well as the filter and compares them so that in case they are different, it returns an error message and does not continue the process. Then it creates the checkerboard array which is multiplied immediately afterwards by the image, as indicated by the methodology of the theory. The result is transferred to the frequency domain where it is filtered by the given filter. The output image is transferred again to the spatial domain and multiplied again by the checkerboard array, to return to its normal form.

The result of image filtering **`demo1Im`** with the filter **`someFreqFilt`** has the following result:

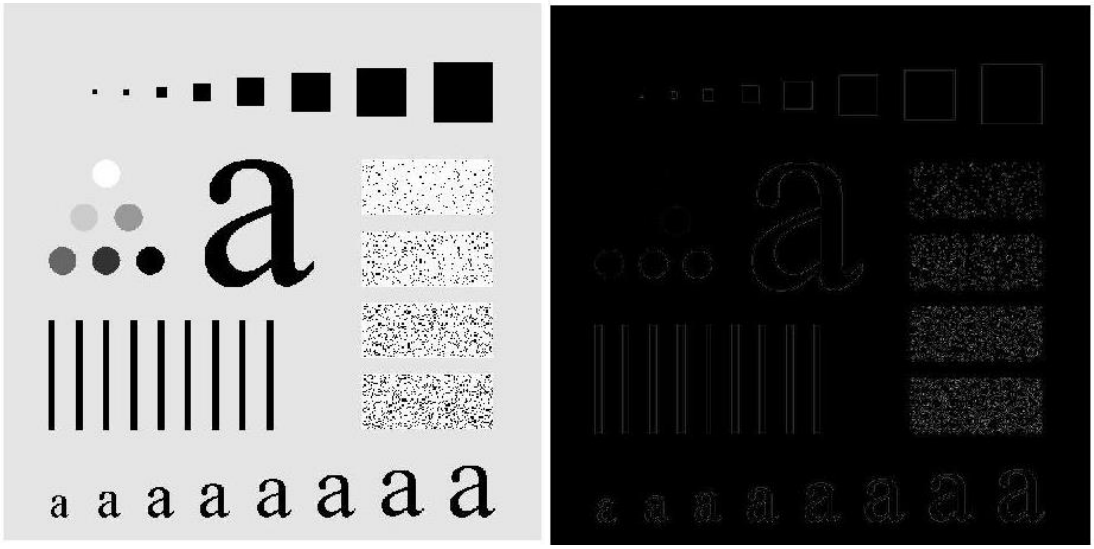


Figure 1.1: The original image given on the right, and the filtered one on the left

What can be clearly observed is that the filtered image is left with only the outlines of the original shapes, which easily leads us to the conclusion that the filter is used for edge detection and could be a low-pass or high-pass filter. Taking the three-dimensional diagram of the filter in MATLAB, we indeed find that it is a high-pass Butterworth type filter.

Section 2

Script Demo2

The given images are loaded and we proceed to build the filters by calling the implemented functions. Here are the images used:

Figure 2.1: The image demo2Im1 on the right, and the image demo2Im2 on the left

Filter Design:

For the design of each filter, the corresponding function was created. Initially, the function code for each type of low-pass filter (Ideal, Butterworth and Gauss) will be explained, and then for their high-pass and band-pass counterparts, which depend directly on the low-pass.

1. For the low-pass Ideal: Initially, the filter matrix is zeroed and a double iteration is run for the upper left quadrant of the matrix. For the rest, the access is unnecessary as there is absolute symmetry towards the center of the matrix. Thus, the formula of the theory is applied: where . The $\boldsymbol{H(u1,u2)} = \begin{cases} 1 & \alpha v\ D(u1,u2) < D0 \\ 0 & \alpha v\ D(u1,u2) > D0 \end{cases}$ $\boldsymbol{D(u1,u2)} = \sqrt{(u1-\frac{M}{2})^2+(u2-\frac{M}{2})^2}$ D0will correspond everywhere to the variable cutoff of the functions.
2. For the low-pass Butterworth: The same logic (of symmetry) is followed, only its formula changes:

   where$H(u1,u2) = \frac{1}{1+D/D0^{2n}} D(u1,u2) = \sqrt{(u1-\frac{M}{2})^2+(u2-\frac{M}{2})^2}$
3. For the low-pass Gaussian: Similarly the same procedure with a different formula:

   $H(u1,u2) = e^{-D^2/2\sigma^2}$ where$D(u1,u2) = \sqrt{(u1-\frac{M}{2})^2+(u2-\frac{M}{2})^2}$
4. For the Ideal high-pass: A corresponding low-pass is created and subtracted from the unit.
5. For the Butterworth high pass: Similarly
6. For the high-pass Gaussian: Similarly
7. For the Ideal bandpass: Two Ideal lowpass filters are created, the "narrow" one which will have the low cutoff frequency (or cutOff) and the "wide" one that will have the highest cutoff. The "narrow" is removed from the "wide" and they form a corresponding bandpass.
8. For the Butterworth bandpass: Similarly
9. For the Gaussian bandpass: Similarly

Here are the filters implemented in a 3D diagram:

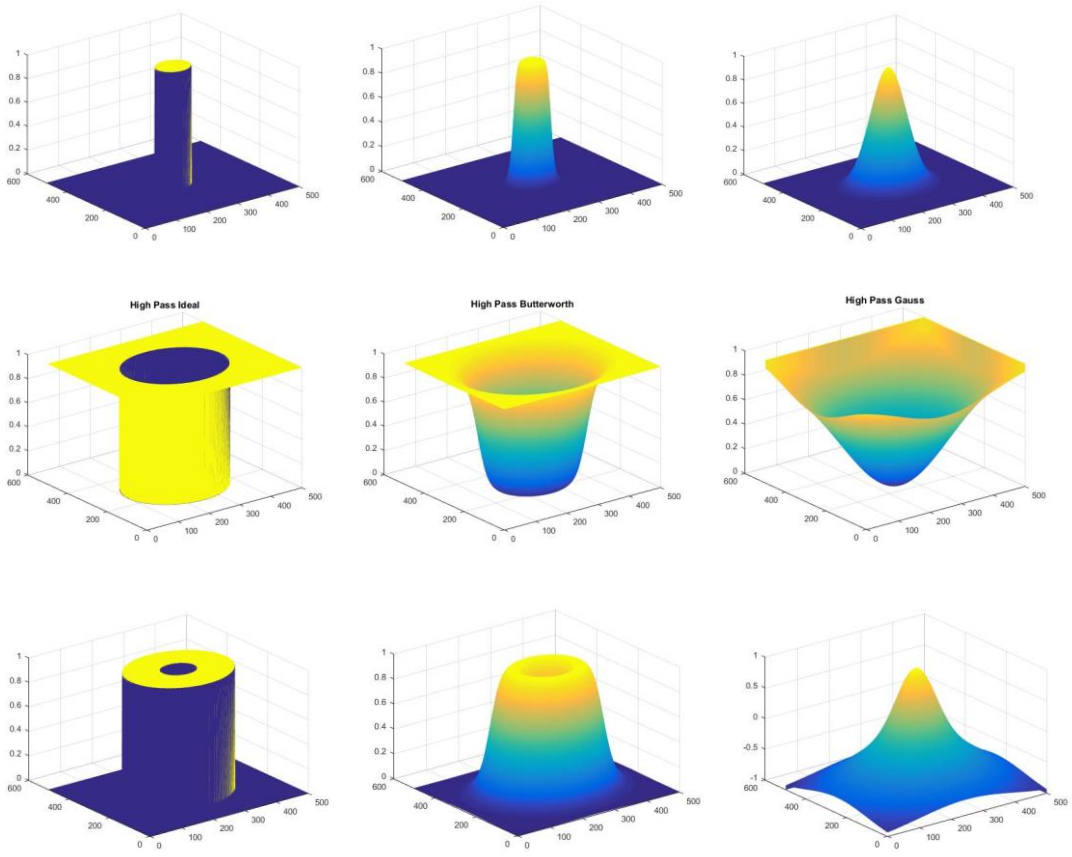


Figure 2.2: The filters in each row are Ideal, Butterworth and Gauss and in each column are low-pass, high-pass and band-pass.

Finally, in the script Demo2, the function myFilterFreq.m is applied 9 times for the first image and 9 for the second, and the corresponding filtered images are output:

Low Pass Ideal - image 1

Low Pass Ideal - image 2

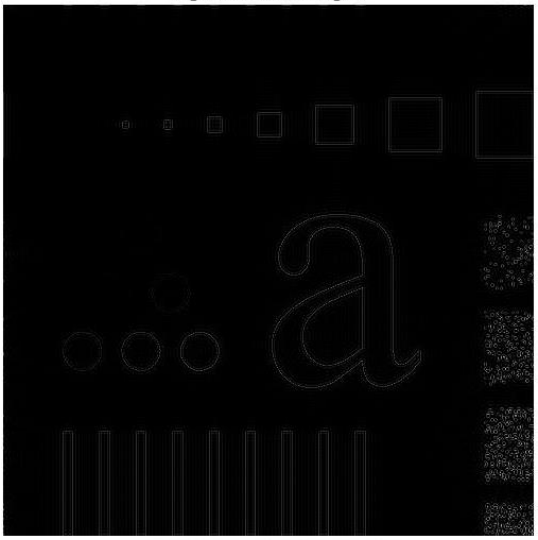
Low Pass Butterworth - image 1

Low Pass Butterworth - image 2

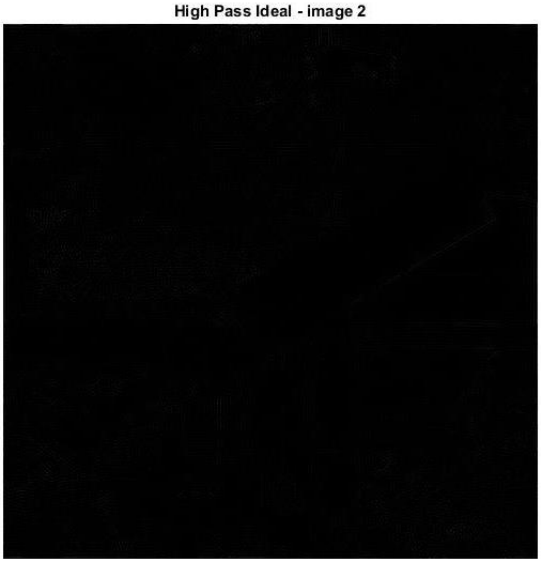
Low Pass Gauss - image 1

Low Pass Gauss - image 2

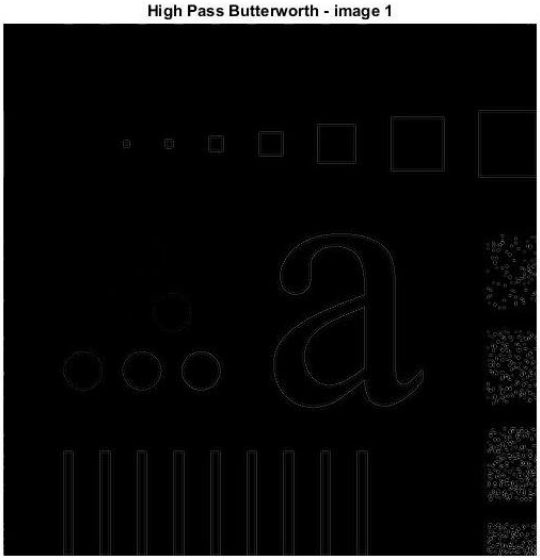
High Pass Ideal - image 1



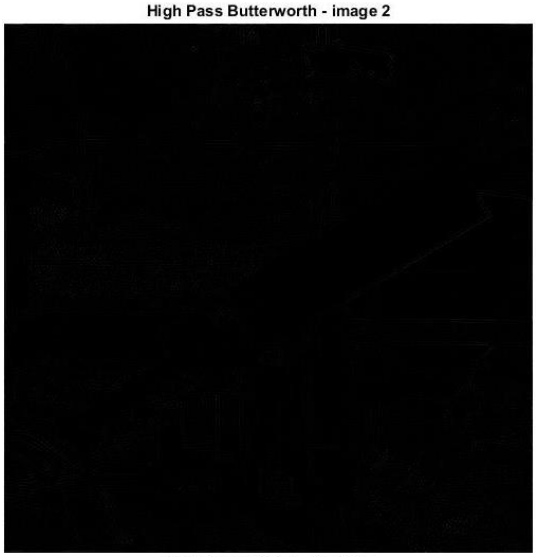
High Pass Ideal - image 2



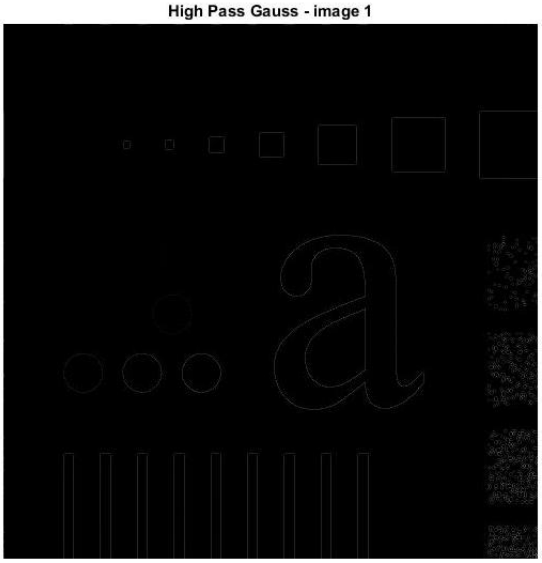
High Pass Butterworth - image 1



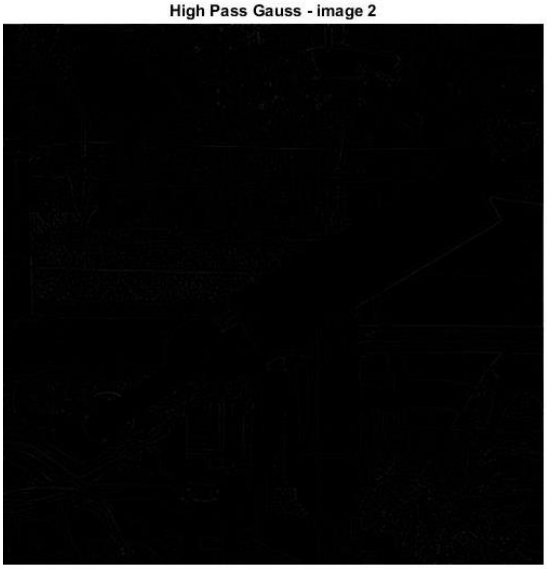
High Pass Butterworth - image 2



High Pass Gauss - image 1



High Pass Gauss - image 2



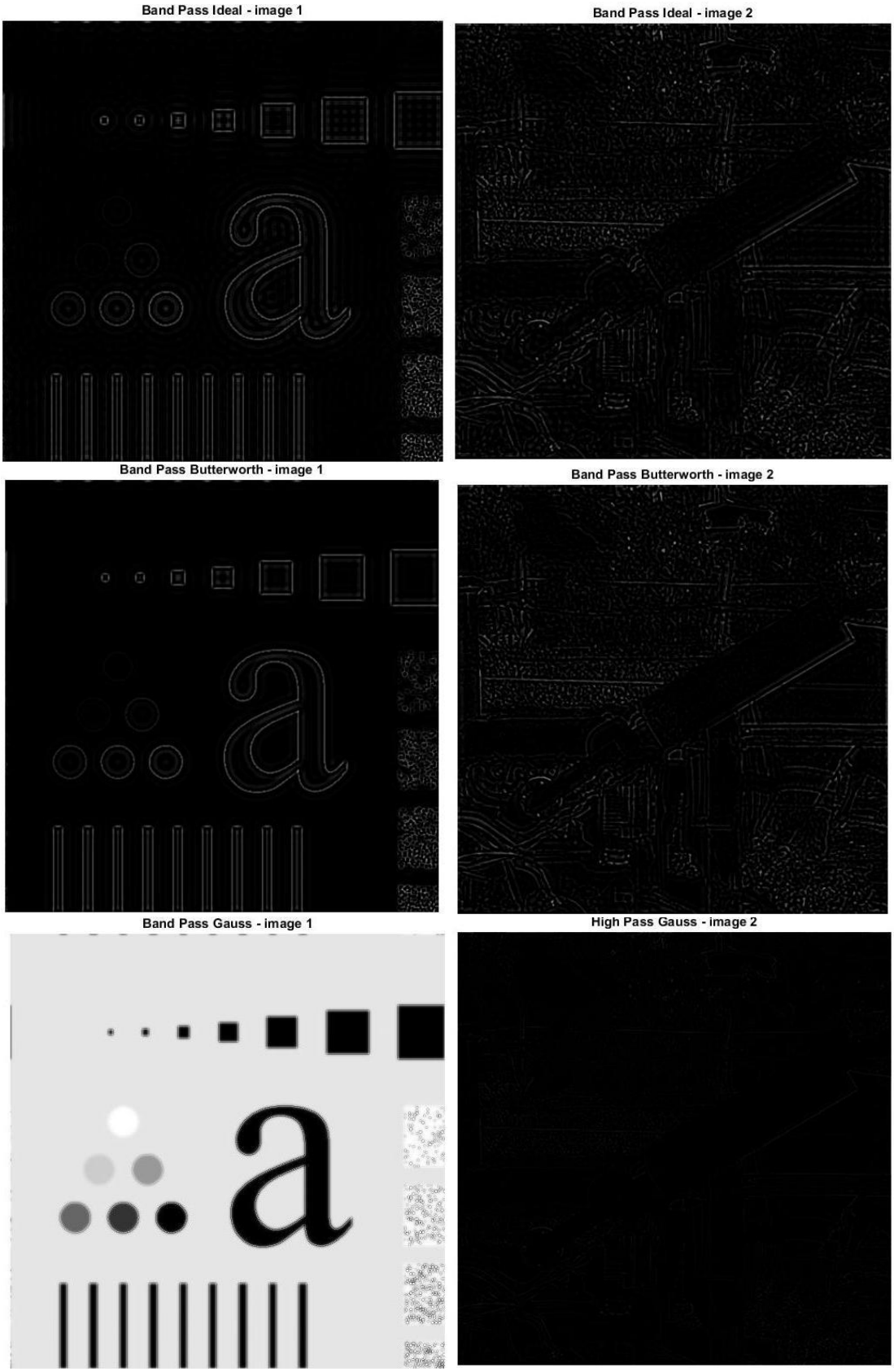


Figure 2.3: The filtered images on the right, demo2Im1, and on the left, demo2Im2, with filtering order from low-pass, high-pass, and band-pass, and individually from Ideal, Butterworth, and Gauss.

Section 3

Script Demo3

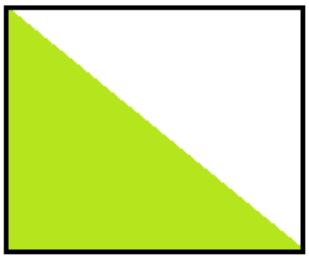
The given images are loaded and we proceed to build the filters by calling the implemented functions. The images used are the same as those in the previous section.

Directional Filters:

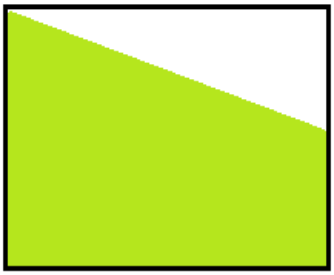
Initially, an explanation will be given of the operation of the directional low-pass Ideal filter and exactly the same procedure will apply to the rest with the exception of the last command of the function code.

In the code, first the filter dimension is checked to be an even number to facilitate calculations. Otherwise, it is simply modified to make theMeven number. Then, the function excludes the possibility of opening the filterf be greater than or equal to 90 degrees. Then the angles are also checked I and θ-ϕ/2(angle1) and θ+ϕ/2(angle2) to belong within the space [0.360). The variables flag1 and flag2 take such a value as to indicate the quadrant to which the angles belong angle1 and angle2 respectively.

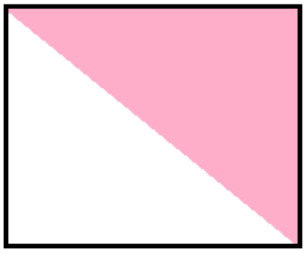
Afterwards, the construction of two masks (M/2xM/2) follows. One for each corner so that in the end they are joined (multiplied) and create the table that will contain the corner with aces inside it and zeros outside it. For this purpose in creating a mask, the value of the corner is checked sequentially (in the first case) angle1and a similar procedure is followed. If the angle takes a value of 0,90,180 or 270 then the mask is a table of units (1 in all cells). If it takes values that are multiples of 45 then a triangular table is created with half having aces (from the cells that create the angle onwards, i.e. in order of increasing the angle) and the other half zeros.



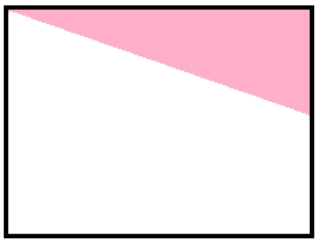
In the different case where the angle takes any other value, a table is created that will form an angle in space(270.0) which is however adapted to each case. The angle is projected onto(0.90)and the corresponding angle in rad. Depending on whether the new angle (newangle1) is greater than 45 degrees or greater, a different variation of the procedure is followed. For newangle1<45calculates the step by which the column counter will be increased based on the formula **j = i tanϕ**. Then for each row the cells belonging to the columns covered by the step of j are calculated. The corresponding procedure is followed for newangle1>45 by reversing i and j. Then, the mask is filled with aces from the cells where the angle is defined and down (i.e. in a decreasing direction, because below, as we see, the mask will be mapped).



Finally, after the vertical inversion of the mask, it is directly adjusted for an initial angle belonging to the1stquadrant. However, if the original angle belongs to the2ndquadrant becomes a clockwise rotation of the mask once, if it belongs to 3^{rd} the mask is rotated twice and if it belongs to the 4^{th} . The mask is rotated three times, resulting in the correct direction being maintained in each case.



The construction of the mask for the corner angle2 requires exactly the same process, with some small but noticeable changes. If the angle coincides with an axis, then instead of an entire table of units, a mask is created with zeros everywhere except for one row or column of aces, which will indicate the end of the angle we are trying toto form. Then if the angle is an integer multiple of 45, a triangular matrix is created again, only the filling with aces is done complementary to that for the mask of the first angle. Finally, if the angle takes any other value, then it becomes exactly the same procedure as above, with the addition that after filling with aces, the mask takes exactly the opposite values (where 0 gets 1, and where 1 gets 0). This procedure also avoids the possibility of overlapping in the merge with other directional filters that will have been formed for an angle right next to the existing one.



Finally, depending on which quadrant the angles belong to angle1 and angle2 (which cannot be more than one quadrant apart, since a restriction was taken for the angle opening to be less than 90 degrees) the masks are joined (through multiplication, resulting in the aces surviving only in the positions where there were aces in both tables), and finally stored in the correct part of the filter, along with their symmetrical ones.

This procedure is followed to create any directional filter. To determine its type and category (low-pass, high-pass, band-pass and Ideal, Butterworth and Gauss) it is simply multiplied by the appropriate simple filter created in the previous section of the work.

At this point let's look at the diagram of some filter examples:

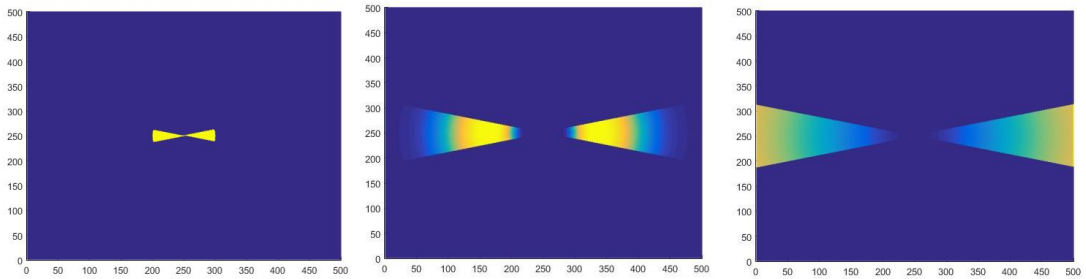


Figure 3.1: Directional filters with ϑ=0 and φ=30. On the left is the Ideal low-pass, in the middle is the Butterworth band-pass, and on the right is the Gauss high-pass.

Then in the Demo2 script, for each filter, its 6 individual directional components are called and combined into one, as indicated by the pronunciation.

~ Note that when combining the individual directional filters to create a single one, there is no overlap except in the center of the filter where the four central cells take the value 2 instead of 1. For this reason, care is taken in the program to find the positions of the cells in each of these filters where the value 1 is exceeded and to return them to unity.~

Finally, in the script Demo2, the function myFilterFreq.m is applied 9 times for the first image and 9 for the second, and the corresponding filtered images are output:

What is generally observed is that each single directional filter causes the same effect on the image as the corresponding simple one.

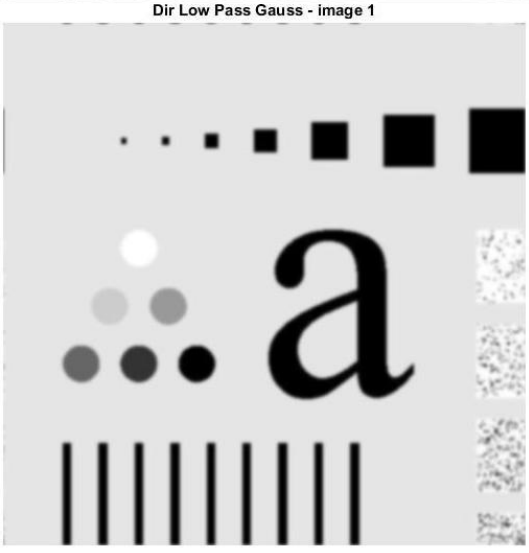
Dir Low Pass Ideal - image 1

Dir Low Pass Ideal - image 2

Dir Low Pass Butterworth - image 1

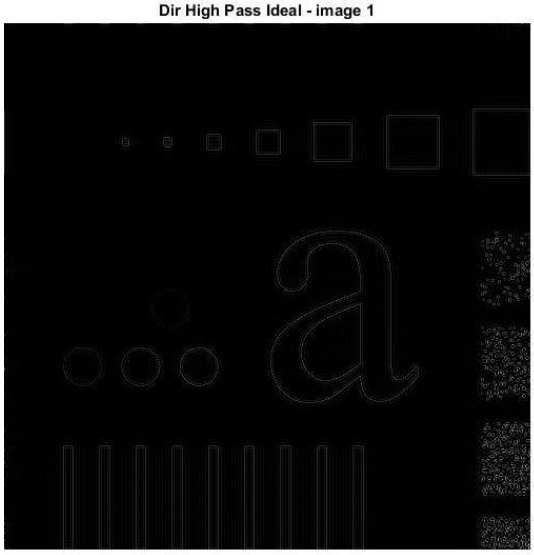
Dir Low Pass Butterworth - image 2

Dir Low Pass Gauss - image 1

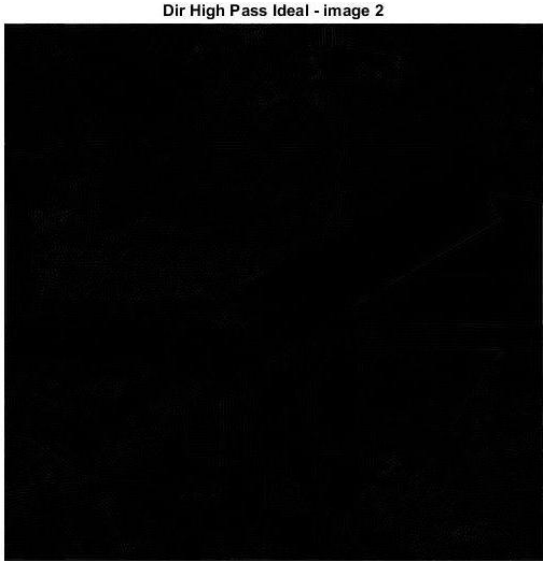


Dir Low Pass Gauss - image 2

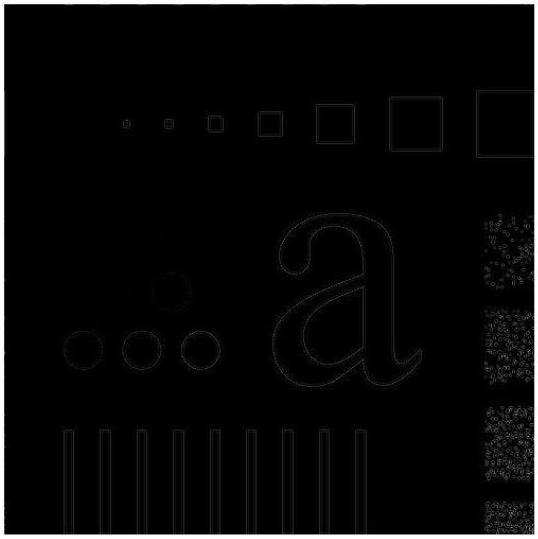
Dir High Pass Ideal - image 1



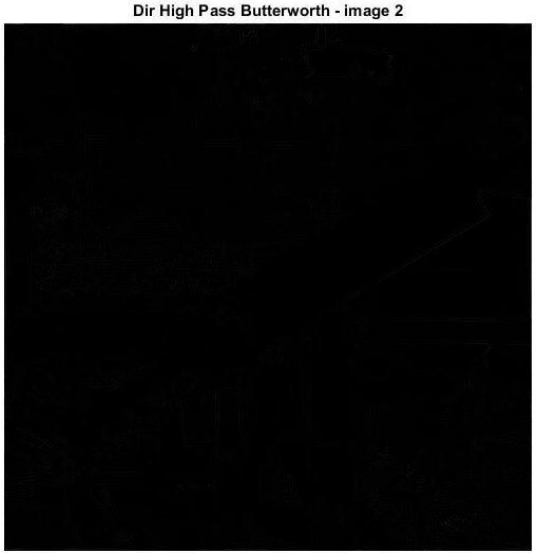
Dir High Pass Ideal - image 2



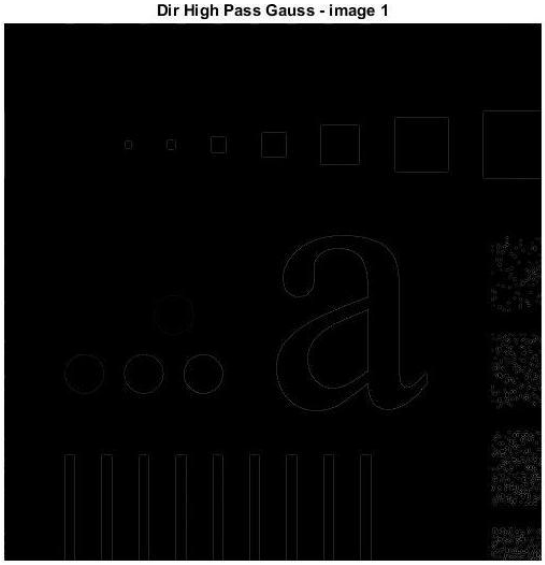
Dir High Pass Butterworth - image 1



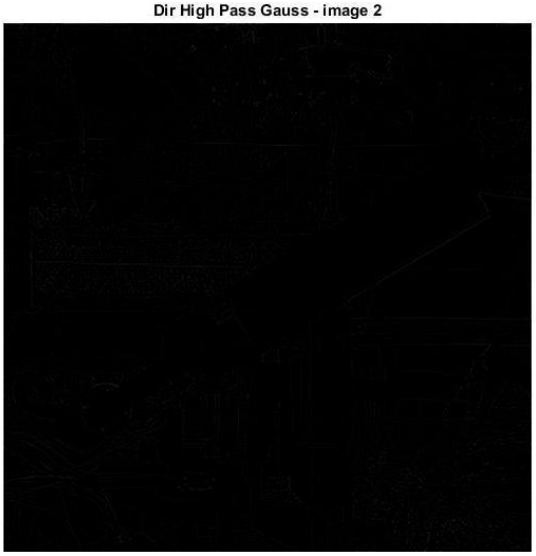
Dir High Pass Butterworth - image 2



Dir High Pass Gauss - image 1



Dir High Pass Gauss - image 2



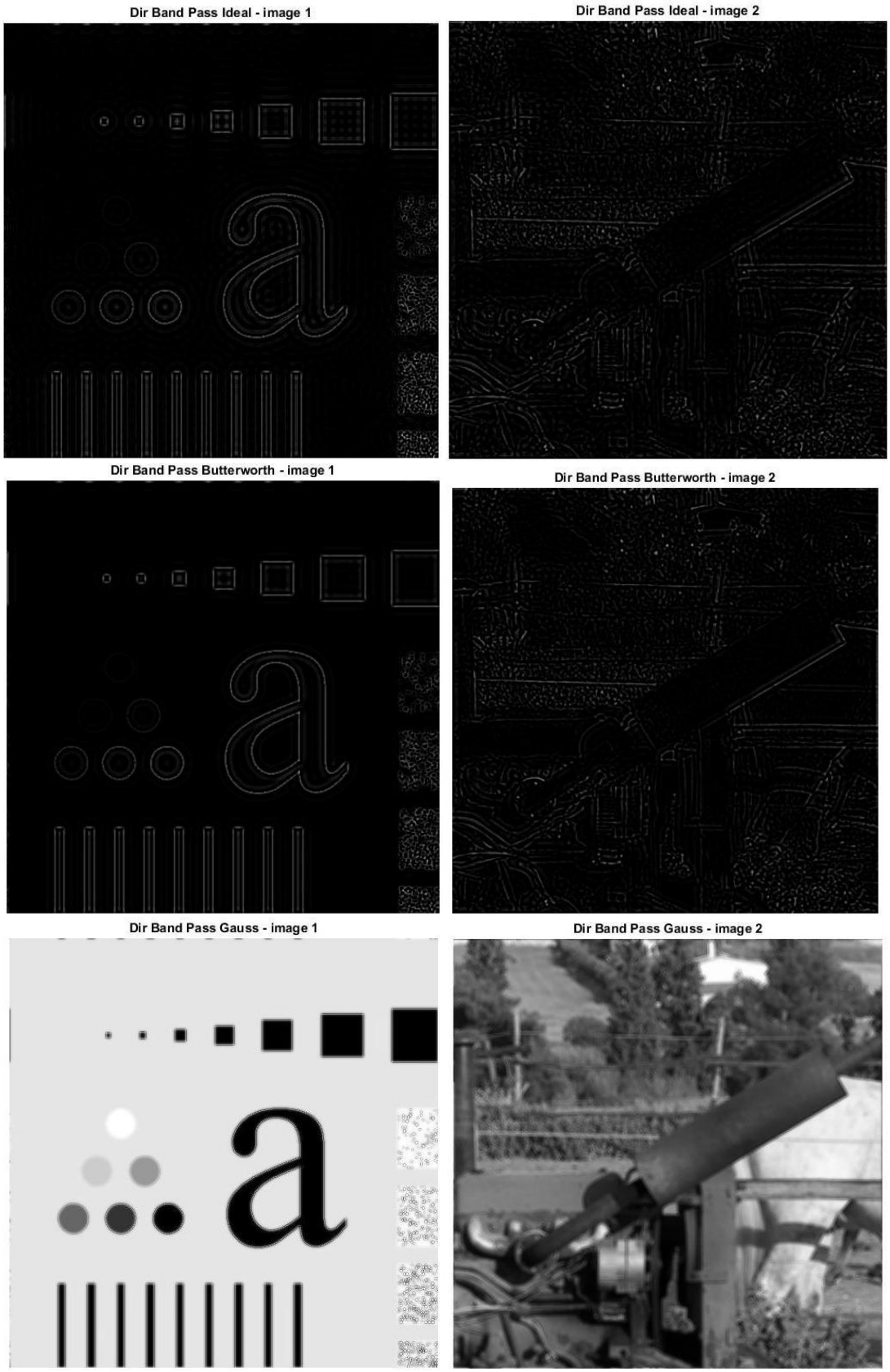


Figure 3.2: The filtered images on the right, demo2Im1, and on the left, demo2Im2, with a directional filtering sequence of low-pass, high-pass, and band-pass, and individual ones of Ideal, Butterworth, and Gauss.