Question 1: Image Filtering

a) Apply a 3×3 median filter to the 5×5 image below and write down the resulting 5×5 image. Then apply a uniform 3×3 smoothing filter (i.e., one that has the same value in every cell) to the original 5×5 image and again write down the resulting 5×5 image.

<table>
<thead>
<tr>
<th>0</th>
<th>150</th>
<th>0</th>
<th>150</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>150</td>
<td>0</td>
<td>150</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>150</td>
<td>0</td>
<td>150</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>150</td>
<td>0</td>
<td>150</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>150</td>
<td>0</td>
<td>150</td>
<td>0</td>
</tr>
</tbody>
</table>

Median filter output:

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>150</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>150</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>150</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Smoothing filter output:

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>50</td>
<td>100</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>50</td>
<td>100</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>50</td>
<td>100</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

b) If you want to blur an image (i.e., make it look like it were out of focus), which of the two filters would you use? Explain why.
Being out of focus means that the light rays that are reflected by the same point in space hit the retina or camera sensor in slightly varying positions rather than in the same position, as it occurs in a sharp image. This means that the intensity of each pixel does not only depend on the intensity of that point in space but, because of this inaccuracy, is a weighted mean of the intensities near that point. This kind of averaging is what a smoothing filter does. A median filter, on the other hand, does not perform averaging but the computation of local medians, which often do not smooth contours but can modify their shape.

**Question 2: Streamlining Sobel**

As you know, for the Sobel filter we define edge magnitude as the square root of the sum of the squared outputs of the horizontal and vertical filters. Now assume that we simplify this definition a bit – edge magnitude is now computed as the sum of the outputs of the two filters.

(a) How could we then make edge detection more efficient by using only one filter instead of two while obtaining exactly the same result for edge magnitude? Determine that filter.

Since convolution is a linear operation, instead of applying the horizontal and vertical filters separately and then add the results, we can simply add the values in the corresponding cells of the two filters. Then a single convolution of our image with the resulting filter will result in the newly defined edge magnitude values.

\[
\begin{bmatrix}
1 & 0 & -1 \\
2 & 0 & -2 \\
1 & 0 & -1
\end{bmatrix}
+ \begin{bmatrix}
1 & 2 & 1 \\
0 & 0 & 0 \\
-1 & -2 & -1
\end{bmatrix}
= \begin{bmatrix}
2 & 2 & 0 \\
2 & 0 & -2 \\
0 & -2 & -2
\end{bmatrix}
\]

(b) Unfortunately, there is a problem with detecting edges using this definition of edge magnitude and the filter you developed. What is the problem, and why does it occur?

The problem is that this filter cannot detect edges in certain orientations. For example, if an image looks like this:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1
\end{bmatrix}
\]

Then the same number of cells with values of 2 and (-2) will fall onto 1s and 0s, and thus the overall result is zero, indicating that there is no edge.
The reason for this problem is that the response of the original Sobel filters to an edge is either a large positive value or a large negative value. By squaring them, adding them, and then taking the square root, we create an edge measure that is never negative and increases with greater edge strength.

However, in the new measure we simply add the two outputs, which means that they can neutralize each other. For example, a strongly positive output of the horizontal filter and a strongly negative output of the vertical filter can result in an overall edge strength measure near zero.

**Question 3 (Bonus): A Square in Hough Space**

Let us perform a Hough transform for the detection of straight lines as we discussed in class. This means that we transform the $x$-$y$ image space into an $\alpha$-$d$ output space. Let us use continuous spaces with conventional coordinate systems instead of matrix notation. Consequently, an angle of $0^\circ$ points in the direction of the $x$-axis, and angles increase in counterclockwise direction. If our input image contains the square whose corners are at the indicated coordinates in the diagram on the left, what pattern of maxima would we expect in the output space? Please enter those maxima into the diagram on the right and also indicate their coordinates, for example, $(90^\circ, 4)$.

**Question 4 (required for CS670, bonus for CS470): A Hough Transform Variant**

For some mysterious application, you need to build an application that detects horizontal lines in a picture. These lines are precisely horizontal, but your client does not only want to know in which rows of the image they are located but also where they start and end, i.e., what their leftmost and rightmost pixels are.
How would you set up a Hough transform to accomplish this task? What would be the dimensions of the output space, and what would be the equation for placing votes?

In order to describe such a line, we need to provide its row $u$ and its leftmost and rightmost columns $v_L$ and $v_R$. These are the 3 dimensions of our output space.

For an edge at position $(i, j)$ we cast a vote at $(u, v_L, v_R)$ if $i = u \land v_L \leq j \leq v_R$. 