# Answers to Practice Problems

1. **Color:**

   (a) The tristimulus coordinates for $c$ are just the weighted sum of the separate tristim coordinates:

$$X_c = 2 \times X_1 + X_2$$

$$Y_c = 2 \times Y_1 + Y_2$$

$$Z_c = 2 \times Z_1 + Z_2$$

   and the chromaticity coordinates are obtained by normalizing

$$x_c = \frac{X_c}{X_c + Y_c + Z_c} = \frac{2 \times X_1 + X_2}{2 \times X_1 + X_2 + 2 \times Y_1 + Y_2 + 2 \times Z_1 + Z_2}$$

$$y_c = \frac{Y_c}{X_c + Y_c + Z_c} = \frac{2 \times Y_1 + Y_2}{2 \times X_1 + X_2 + 2 \times Y_1 + Y_2 + 2 \times Z_1 + Z_2}$$

2. **Detecting gaps in lines:**

   (a) The simplest masks you could have given that would be useful for this purpos e with the hit-or-miss transformation are:

$$\begin{matrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{matrix} \qquad \begin{matrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{matrix} \qquad \begin{matrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{matrix} \qquad \begin{matrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{matrix}$$

   For example, the first one would find a place where there is a gap in a horizontal line. If there is a “hit” we consider that a 1-pixel gap has been detected in the center location.

   There are some configurations where these masks won’t work perfectly. For example, if we had a horizontal line crossing a vertical line just above a place where the vertical line has a break, the masks listed would not find that break. Additional masks or “don’t care” states could be added to help deal with various problems of crossing lines.

   (b) Convolution approach:

   If we convolve with the following 4 masks:

$$\begin{matrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{matrix} \qquad \begin{matrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{matrix} \qquad \begin{matrix} 0 & 0 & 1 \\ 0 & -2 & 0 \\ 1 & 0 & 0 \end{matrix} \qquad \begin{matrix} 1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 1 \end{matrix}$$

   then each mask would

   - yield a value of zero if centered on an empty space, or if centered on a pixel of an unbroken 3-pixel segment oriented in the direction favored by the mask

- yield a value of $-2$ when centered on a pixel of an unbroken 3-pixel segment oriented in a direction NOT favored by the mask
- yield a value of $+1$ when placed next to a line of a different direction
- yield a value of $+2$ when centered on a one-pixel gap in a 3-pixel segment oriented in the direction favored by the mask

So the results can be thresholded at $+2$. The method will be confused in various cases (e.g., two parallel lines with a 1-pixel wide gap between them). Again, more complicated masks, or additional masks could be used that would help distinguish between some of these cases.

3. **Filter size effects:**

Consider an image that has stripes of intensity "a" and width 3, alternating with stripes of intensity 0 and width 2. A section of the image might look like this:

00aaa00aaa00aaa00
00aaa00aaa00aaa00
00aaa00aaa00aaa00

If you filter this with an unweighted square spatial averaging filter of size 3x3, you will get a pattern like this (where c = a/3, and b = 2a/3):

ccbabccbabccbabcc
ccbabccbabccbabcc
ccbabccbabccbabcc
So the bright stripes of intensity a have been smeared (blurred) but they are still correctly centered on the same places that they used to be centered.

If instead we used a 5x5 filter, we would get a completely uniform intensity pattern for the whole image, where the uniform intensity has value 3a/5. This is because the width of the filter, 5, is exactly equal to the width of one cycle of the pattern (00aaa). So, when this filter slides across, it always picks up one cycle of the pattern. (Note that you would also get this effect if your filter width were some multiple of 5.)

If instead we used a 7x7 filter, we would get an output:

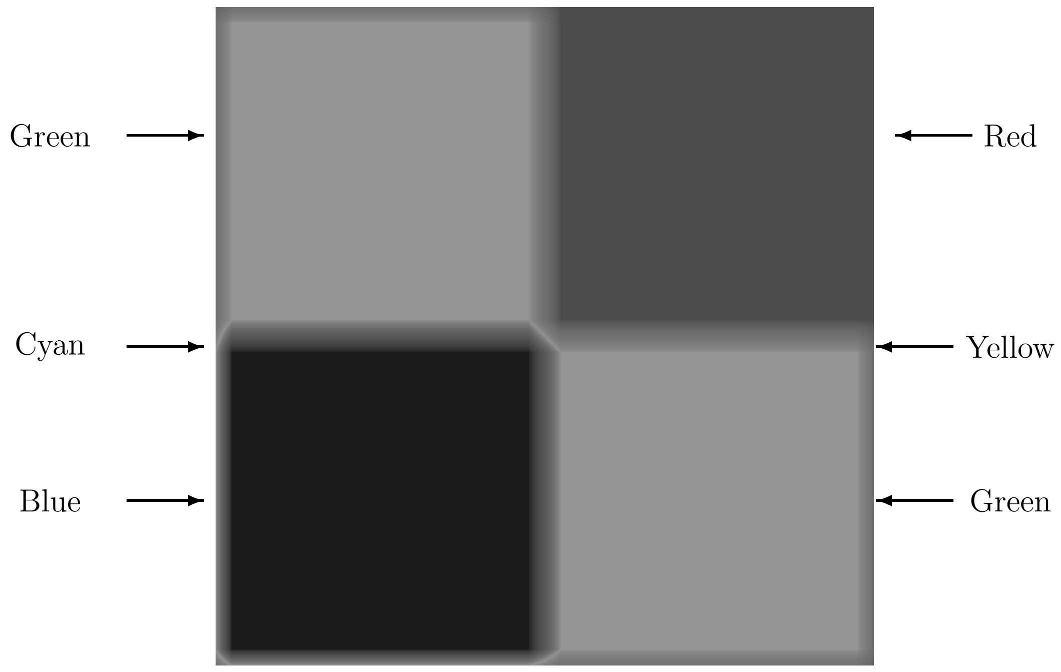
ddefeddefeddefedd
ddefeddefeddefedd
ddefeddefeddefedd
where d = 5a/7, and e = 4a/7, and f = 3a/7. Now again the stripes can be noticed, although they are lower contrast than before. Also, the bright places of the output (value d) are located where there used to be dark parts (value 0). So, it is as though the bright stripes have shifted over. This is an effect called pseudo-resolution. That is, after using the 3x3 filter, we can still correctly resolve the bars in the locations where they are supposed to be, but after using the 7x7 filter, the bars appear to be resolved but in fact they are shifted.

The reason why this happens is the filter, when centered on the dark stripes, in wide enough to encompass some or all of the bright stripes on BOTH sides. Whereas when it is centered on the bright stripe, it can encompass only the ONE bright stripe and the zero valued stripes on both sides, so it actually has a smaller output value.

This is the general idea of what is going on in these pictures.

4. **Color: Hue and Saturation**

(a) When you blur the hue, and then transform back to RGB, the image looks like this:



Between the green and blue regions, there is a strip of width 24 pixels (that is, 12 pixels on each side of the center line) which passes in color from green, through cyan, to blue. Between the red and green regions, there is a strip of width 24 pixels which passes in color from red, through yellow, to green.

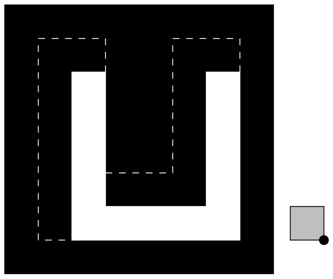
In the center of the image, there is a region where the filter will be encountering all 4 quadrants, and so the result there is going to be a bit more complicated.

Near the boundary of the image, if the image is zero-padded prior to filtering, then there are going to be altered colors at the boundary as well. That is the situation shown in the figure above. But if the boundary were handlded by replicating the first/last row/column outward as many times as needed for the filtering, then there would be no altered color effects at the boundary. In any case, you were not expected to mention this.

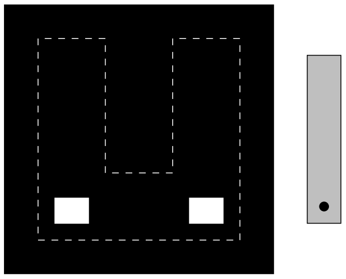
(b) Lastly, when you blur the saturation, there is no effect, because all the 4 quadrants are equally saturated to start with.

5. **Binary image morphology:**

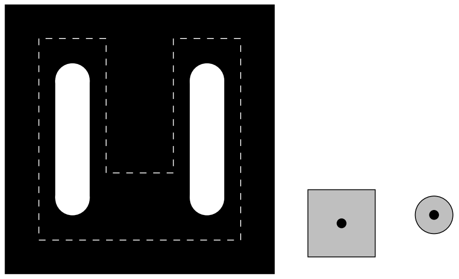
   Solution (a) was obtained by eroding the original set with the structuring element shown. Note that the origin of the structuring element is at the bottom right, represented by a black dot. Solution (b) was obtained by eroding the original set with the tall rectangular structuring element shown. Solution (c) was obtained by first eroding the image shown down to vertical lines using the square-ish structuring element; this result was then dilated with the circular structuring element. Note that the square-ish structuring element might have to be a little bit rectangular, so that it is a little taller than the center section of the U. Otherwise, when you erode there will be a horizontal line across the bottom of the U. But if the structuring element is slightly too tall, then it won't fit there, and the erosion will produce just two vertical lines, which can then be dilated with the circular element.



***(a)***



***(b)***



***(c)***

6. **Distance transformations:**

   Consider that the general 3x3 chamfer matrix has values a and b, where b is the distance that propagates diagonally. If we take a=1 for all these 3 DTs, then CityBlock has b=$\infty$, Chessboard has b=1, and Chamfer3-4 has b=1.333.

   On this problem, some people take the simplistic view that the b value can be directly compared with $\sqrt{2}$. This would lead us to say that Chessboard and Chamfer3-4 produce smaller values than EDT, and CityBlock is larger. This is not right. If this were right, we could just take b=$\sqrt{2}$ and get the true Euclidean distance without doing a global calculation.

   For CityBlock, the distance between pixel grid points $(x_1, y_1)$ and $(x_2, y_2)$ is

   $|x_1 - x_2| + |y_1 - y_2|$

   and this is always greater than or equal to the EDT: $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

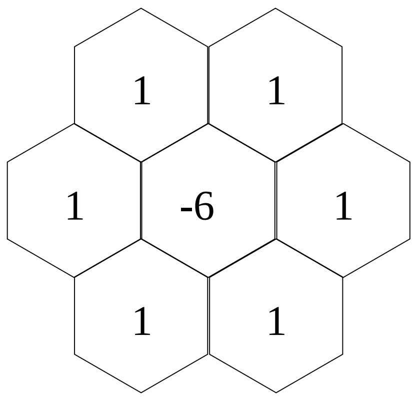
   For Chessboard, the distance between the points is:
   $max\{|x_1 - x_2|, |y_1 - y_2|\}$

   and this is always less than or equal to the EDT.

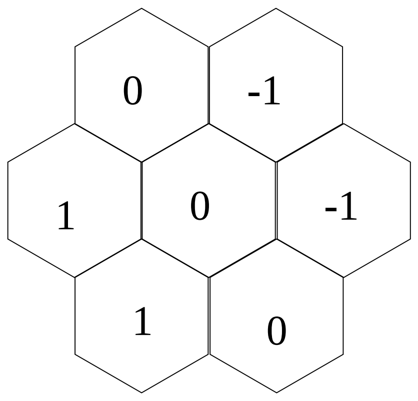
   Lastly, Chamfer3-4 is sometimes less and sometimes greater. Consider a pixel that is one up and one to the right. EDT is $\sqrt{2}$ and Chamfer3-4 is 1.333 which is less. However, if we consider a pixel that is one up and two to the right, EDT is $\sqrt{5}$ and Chamfer3-4 is 2.333 which is larger.

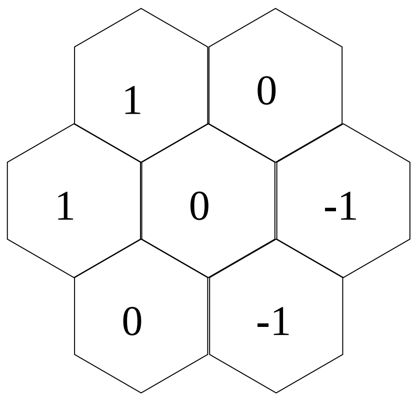
7. **Edge detection:**

   The Laplacian operator on a hexagonal grid is best defined like this:

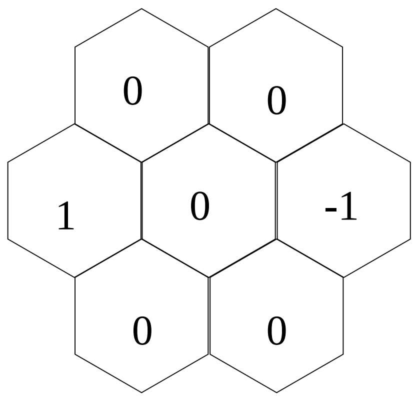


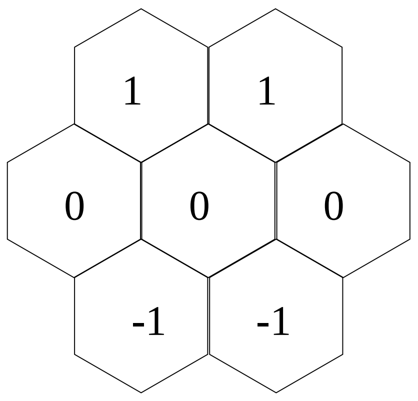
   The first derivative edge detectors can be defined a number of different ways on the hexagonal grid. For example, the following masks provide orthogonal first derivative operators that are analogous to the Prewitt operators, although they are for the diagonal directions:





   Another possibility is like this, in which we still have orthogonal operators, now for the horizontal and vertical directions, although they no longer are rotated versions of each other:





   And there are a few other sensible possibilities too.