Homework #1
SOLUTION

1. In this problem, you will determine the finest detail that the human eye can discern at a set distance. Assume the pinhole camera model in figure 2.3 of the text where the distance between the pinhole and the retina along the visual axis is 17mm. Assume that the density of cones in the fovea is 150,000 elements per mm^2 and that the cones are arranged in a grid with no spacing between them.

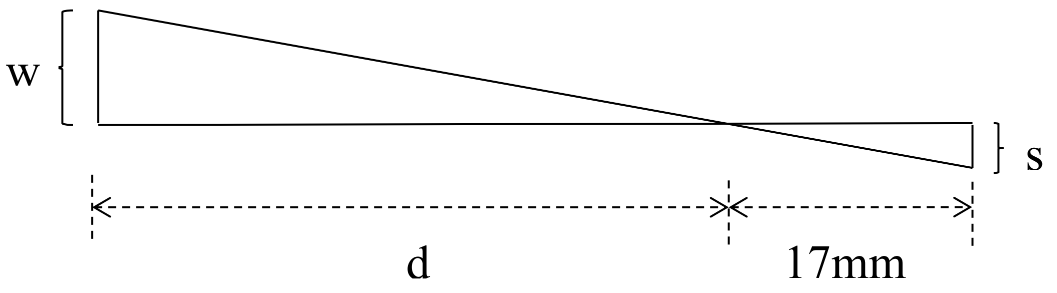
   Suppose you are looking at a scene with alternating black and white lines of equal width. Assume that you are able to discern the individual lines up to the point where the image of a line on your retina is smaller than a single cone. That is, when the image of a line is smaller than a cone, you can no longer tell it apart from an adjacent line.

   Calculate the width of the smallest line you can discern when the scene is:

   a) 0.2 meters from your eye (from the pinhole). (Note, due some simplifying assumptions, you will probably get a result which seems smaller than you expect.)

   SOLUTION:
   We first compute how large a single cone is. A density of 150,000 cones per mm^2 corresponds to a 1mm by 1mm grid of $\sqrt{150{,}000}$ or 387 cones per side. Thus, each cone measures approximately $1mm/387$ or $2.58\text{x}10^{-6}$m along each dimension.

   Figure 2.3 models the human eye as the following pinhole camera (the diagram below is a bit different from that in figure 2.3 but they are really the same):



   Where *d* is the distance of an object, *w* is the size of the object, and *s* is the size of the image the object makes on the retina. From similar triangles,

$$\frac{w}{d} = \frac{s}{1.7 \times 10^{-2} m}$$

In this first case, d=0.2m. In order to see an individual line, we need

$$s \geq 2.58 \times 10^{-6} m.$$

Substituting, we get

$$w \geq 3.04 \times 10^{-5} m.$$

That is, the lines must be larger than approximately 0.03mm in width to be discerned at 0.2m.

b) 100 meters from your eye.

SOLUTION:
Now, d=100m and we must have

$$w \geq 1.52 \times 10^{-2} m.$$

That is, the lines must be larger than approximately 1.52cm in width to be discerned at 100m.

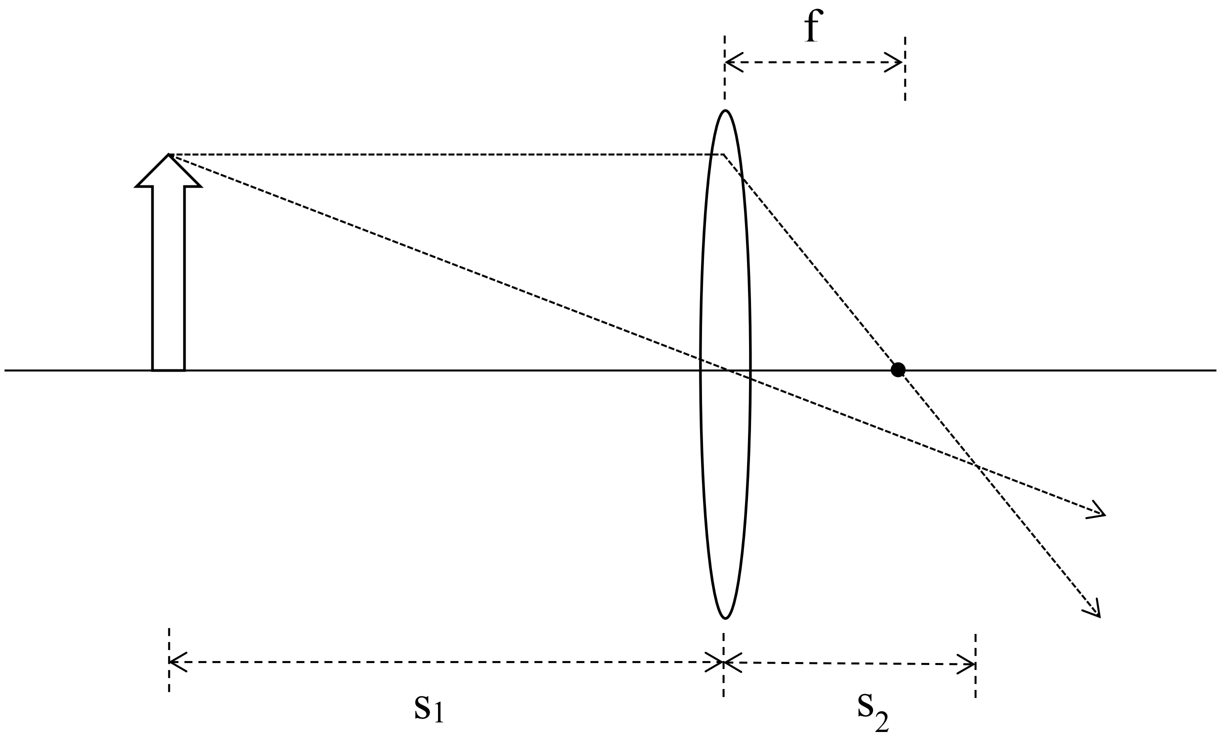
2. In this problem, you will determine the correct distance the imaging plane should be from the lens for an object at a certain distance to be in focus.

   Assume the thin lens model discussed in lecture for a camera with a focal length of 35mm. An object in the scene will be in focus when all the rays from the object that pass through the lens converge at the same point on the imaging plane.

   What should be the distance from the lens to the imaging plane for an object that is:

   a) 1m from the camera?

   SOLUTION:



Shown above is the geometry of a thin lens model. As derived in class, the relation between the focal length *f*, the distance of the object s_1, and the distance of where the rays of light converge s_2 is

$$\frac{1}{f} = \frac{1}{s_1} + \frac{1}{s_2}$$

We have f=35mm. For this first case, s_1=1m so from

$$s_2 = (f^{-1} - s_1^{-1})^{-1}$$

we have s_2=0.0362m or 36.2mm.

b) 100m from the camera?

SOLUTION:
Now, s_1=100m and so s_2=0.0350m or 35.0mm. Not much difference from the first case.

3. In this problem, you will calculate the cost of transmitting various forms of multimedia across a mobile data channel.

   Assume that your mobile data plan costs you \$30 for 2GB (where 1GB is 1,024MB, 1MB is 1,024 KB, and 1KB is 1,024 bytes). Assume, also, that you pay proportionally. That is, if your usage is not an increment of 2GB then you pay only for the fraction that you use (whether it is less or more than 2GB).

   a) Calculate how much it would cost for you to download all of Shakespeare's plays where the total word count of all his plays is 835,997. Assume the average word length is five letters and that you represent each letter using a byte (we are ignoring spaces, punctuation, and formatting).

   SOLUTION:
   We can compute the cost per byte as

$$\frac{\$30}{2GB} \times \frac{1GB}{1{,}024MB} \times \frac{1MB}{1{,}024KB} \times \frac{1KB}{1{,}024bytes} = \$1.40 \times 10^{-8}\ per\ byte$$

   At an average of five letters per word, there are 4,179,985 letters in all his plays.

   So, the cost to download all of Shakespeare's plays is approximately

$$\$5.85 \times 10^{-2}$$

   or, about six cents.

   b) Calculate how much it would cost you to download a single image from a Canon EOS 5D Mark III DSLR camera without any compression.

   This camera produces 22.3 mega-pixel images where

$$1\ mega - pixel = 1{,}024 \times 1{,}024\ pixels$$

   and each pixel is represented using three bytes, one for each of the red, green, and blue color channels.

   SOLUTION:
   We first compute how many bytes are required to represent each image:

$$\frac{22.3\, mega-pixels}{1\, image} \times \frac{1{,}024 \times 1{,}024\, pixels}{1\, mega-pixel} \times \frac{3\, bytes}{1\, pixel} = 7.01 \times 10^{7}\, bytes/im$$

Now, multiplying by the cost per byte, the cost for downloading a single image is approximately

$$\$9.8 \times 10^{-1}$$

or, about one dollar.

c) Calculate how much it would cost to download a one-hour ultra high-definition 4k video without compression.

   A 4k video has 60 frames per second where each frame measures 3,840 by 2,150 pixels. Again, assume each pixel is represented using three bytes.

SOLUTION:
We first compute how many bytes are required to represent the video:

$$\frac{60 \times 60\, seconds}{1\, video} \times \frac{60\, frames}{1\, second} \times \frac{3{,}840 \times 2{,}150\, pixels}{1\, frame} \times \frac{3\, bytes}{1\, pixel} = 5.35 \times 10^{12}\, bytes/video$$

Now, multiplying by the cost per byte, the cost for downloading the video is approximately

$$\$7.49 \times 10^{4}$$

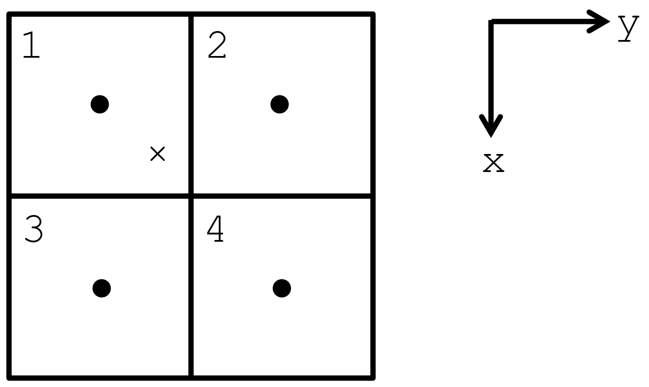
or, about \$75,000. It's a good thing we have compression!

Homework #2
SOLUTION

PROBLEM 1:

This problem investigates nearest neighbor and bilinear interpolation. For simplicity, we will focus on estimating the image intensity at a single location. Interpolation is used when transforming an image through resizing, rotating, etc. in which case, the image intensity will need to be estimated at a number of locations.

Consider the diagram below of four pixels



where the dots (●) represent the locations where we know the image intensity and the × represents the location where we would like to estimate the image intensity. By convention, the vertical axis is the x-axis and the horizontal axis is the y-axis.

Suppose the four pixels are at the following locations (indicated by (*x*,*y*)) and have the following intensity values (indicated by *p*)

$$
\begin{aligned}
(x_1, y_1) &= (4,10) \quad & p_1 &= 100 \\
(x_2, y_2) &= (4,11) \quad & p_2 &= 107 \\
(x_3, y_3) &= (5,10) \quad & p_3 &= 120 \\
(x_4, y_4) &= (5,11) \quad & p_4 &= 130
\end{aligned}
$$

and that we would like to estimate the image intensity at a fifth location

$$(x_5, y_5) = (4.3,10.4)$$

That is, we want to estimate p_5.

a) Provide an estimate for p_5 using nearest neighbor interpolation.

SOLUTION:

Nearest neighbor interpolation assumes the digital image is a sampled version of a continuous image that has constant intensity from a pixel center to the boundaries of the adjacent pixels. Therefore, to estimate the value at a particular location, we first determine which pixel boundary it falls in and then assign the value from that pixel.

To do this, we simply use the value of the nearest pixel (and thus the name of the method). In our case, the nearest pixel to location $\times$ is pixel 1 which has a value of 100. So,

$$p_5 = 100.$$

b) Provide an estimate for p_5 using bilinear interpolation. Round your value to the nearest integer. You can use either of the two methods discussed in lecture. (You might want to use both methods to check your answer.)

SOLUTION:

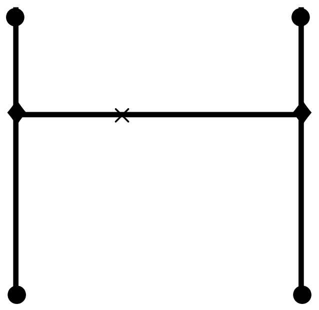
Nearest neighbor interpolation assumes the digital image is a sampled version of a continuous image with a bilinear surface. That is, the surface varies linearly along each of the x and y axes, or, equivalently, has the parametric form:

$$f(x,y) = ax + by + cxy + d \quad .$$

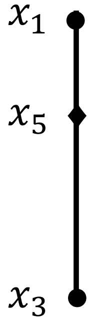
Method 1:

The first method presented in lecture performs linear interpolation in stages, first along one dimension and then the other. The order does not matter.

We will first interpolate along the x-axis to estimate the values at the locations indicated by the diamonds (♦) in the figure below and then interpolate along the y-axis to estimate the value at the location $\times$, which is what we want.

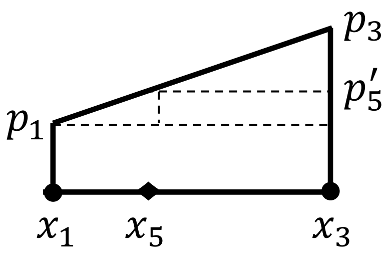


We have the following linear situation for pixels 1 and 3



where location x_1 has value p_1, location x_3 has value p_3, and we want to estimate the value at location x_5 which we will denote as p_5' .

Looking at this “sideways”, where height is the image intensity, we have



From similar triangles, we have

$$\frac{p_5' - p_1}{x_5 - x_1} = \frac{p_3 - p_1}{x_3 - x_1}$$

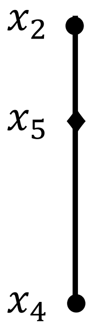
or,

$$p_5' = (p_3 - p_1)\frac{(x_5 - x_1)}{(x_3 - x_1)} + p_1 \,.$$

Substituting for the known values, $x_1 = 4$, $x_3 = 5$, $x_5 = 4.3$, $p_1 = 100$, and $p_3 = 120$, we get

$$p_5' = 106 \,.$$

Repeating for pixels 2 and 4



where location x_2 has value p_2, location x_4 has value p_4, and we want to again estimate the value at location x_5 which we will denote as p_5'' .

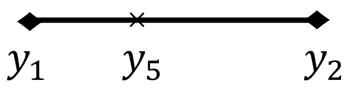
Using similar triangles, we end up with

$$p_5'' = (p_4 - p_2)\frac{(x_5 - x_2)}{(x_4 - x_2)} + p_{21} \,.$$

Substituting for the known values, $x_2 = 4$, $x_4 = 5$, $x_5 = 4.3$, $p_2 = 107$, and $p_4 = 130$, we get

$$p_5'' = 113.9 \,.$$

We now have the following linear situation for the two locations that we just estimated the value at (represented by diamonds):



Using similar triangles, we end up with

$$p_5 = (p_5'' - p_5')\frac{(y_5 - y_1)}{(y_2 - y_1)} + p_5' \,.$$

Substituting for the known values, $y_1(= y_3) = 10$, $y_2(= y_4) = 11$, $y_5 = 10.4$, $p_5' = 106$, and $p_5'' = 113.9$, we get

$$p_5 = 109.16$$

or, rounding to the nearest integer

$$p_5 = 109\,.$$

Method 2:

Here, we first determine the parameters a, b, c, and d in the bilinear equation

$$f(x, y) = ax + by + cxy + d \quad .$$

We then can simply use this equation to estimate the value at any location.

Assuming the surface specified by the bilinear equation goes through the four locations at which we know the intensity values, we have:

$$\begin{aligned} p_1 &= ax_1 + by_1 + cx_1y_1 + d \\ p_2 &= ax_2 + by_2 + cx_2y_2 + d \\ p_3 &= ax_3 + by_3 + cx_3y_3 + d \\ p_4 &= ax_4 + by_4 + cx_4y_4 + d \end{aligned}$$

All the values in these equations are known except for the parameters a, b, c, and d. This is a system of four equations and four unknowns.

A simple way to solve for the unknowns is to write this system in matrix multiplication form:

$$\begin{bmatrix} x_1 & y_1 & x_1y_1 & 1 \\ x_2 & y_2 & x_2y_2 & 1 \\ x_3 & y_3 & x_3y_3 & 1 \\ x_4 & y_4 & x_4y_4 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$$

or

$$Ag = h\,.$$

We can solve for g

$$g = A^{-1}h\,.$$

In our case,

$$A = \begin{bmatrix} 4 & 10 & 40 & 1 \\ 4 & 11 & 44 & 1 \\ 5 & 10 & 50 & 1 \\ 5 & 11 & 55 & 1 \end{bmatrix}$$

so,

$$A^{-1} = \begin{bmatrix} -11 & 10 & 11 & -10 \\ -5 & 5 & 5 & -4 \\ 1 & -1 & -1 & 1 \\ 55 & -50 & -44 & 40 \end{bmatrix} \quad .$$

(You can compute this matrix inverse in Matlab.)

And,

$$h = \begin{bmatrix} 100 \\ 107 \\ 120 \\ 130 \end{bmatrix}$$

so,

$$g = \begin{bmatrix} -10 \\ -5 \\ 3 \\ 70 \end{bmatrix}$$

or, $a = -10$, $b = -5$, $c = 3$, and $d = 70$.

We now know the bilinear equation

$$f(x, y) = -10x + (-5)y + 3xy + 70$$

and can estimate the intensity at $(x_5, y_5) = (4.3{,}10.4)$ as

$$p_5 = 109.16$$

or, rounding to the nearest integer

$$p_5 = 109 .$$

PROBLEM 2:

SOLUTION:

Let p and q be as shown in Fig. P2.11. Then, (a) S_1 and S_2 are not 4-connected because q is not in the set $N_4(p)$; (b) S_1 and S_2 are 8-connected because q is in the set $N_8(p)$; (c) S_1 and S_2 are m-connected because (i) q is in $N_D(p)$, and (ii) the set $N_4(p) \cap N_4(q)$ is empty.

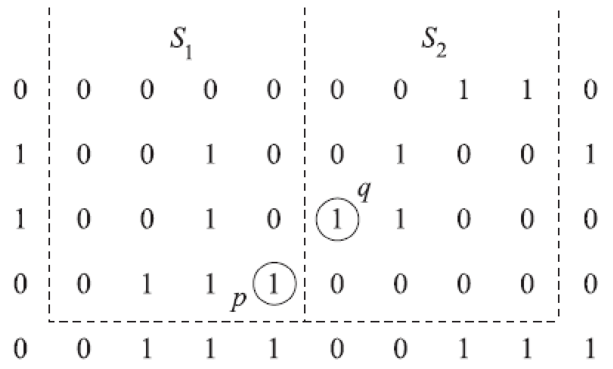


Figure P2.11

PROBLEM 3:

SOLUTION:

(a) When $V = \{0,1\}$, 4-path does not exist between p and q because it is impossible to get from p to q by traveling along points that are both 4-adjacent and also have values from V. Figure P2.15(a) shows this condition; it is not possible to get to q. The shortest 8-path is shown in Fig. P2.15(b); its length is 4. The length of the shortest m- path (shown dashed) is 5. Both of these shortest paths are unique in this case.

(b) One possibility for the shortest 4-path when $V = \{1,2\}$ is shown in Fig. P2.15(c); its length is 6. It is easily verified that another 4-path of the same length exists between p and q. One possibility for the shortest 8-path (it is not unique) is shown in Fig. P2.15(d); its length is 4. The length of a shortest m-path (shown dashed) is 6. This path is not unique.

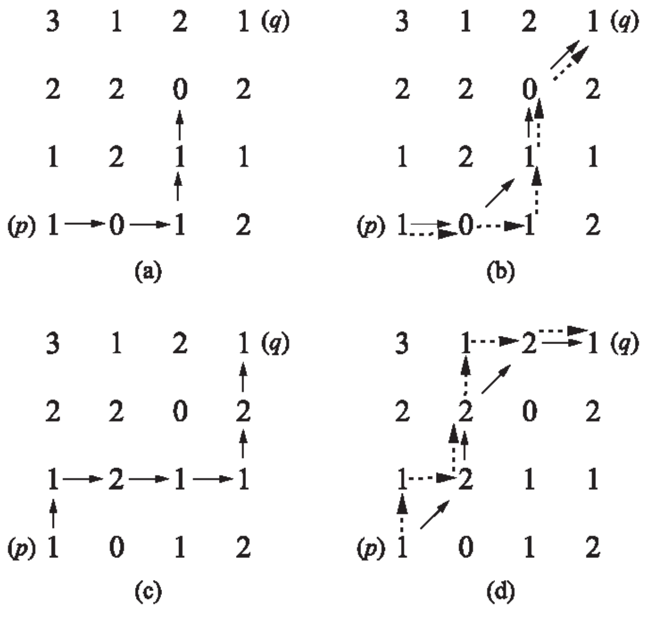


Figure P.2.15

PROBLEM 4:

This problem will help with your project on image resizing.

a) Find the linear mapping from a location m′ in the transformed image to the location x in the original image. That is, derive a function that given m′, computes x. The transformed image measures M′ pixels and the original image measures M pixels. Note that M′ can be greater than or less than M.

Your mapping should have the following form

$$x = t(m') = \frac{A}{B}\left(m' - C\right) + D$$

where the constants A, B, C, D are determined from the following constraints:

- The left boundary of the transformed image should map to the left boundary of the original image.
- The right boundary of the transformed image should map to the right boundary of the original image.
- Locations in between should be mapped linearly (proportionally).

Note that the linear equation above really only has two constants and could be written as

$$x = t(m') = Am' + B .$$

I provided the four-constant version above to help you think about how to incorporate the constraints (that form helps me, at least). You can use either form in your answer.

Note that x typically won't be integer valued (that is, it won't fall on a pixel location in the original image) even if m′ is integer valued.

SOLUTION:

$$x = t(m') = \frac{M}{M'}\left(m' - 0.5\right) + 0.5$$

so that

$$t(0.5) = 0.5$$

and

$$t(M' + 0.5) = M + 0.5 .$$

$t(\cdot)$ can also be written as

$$x = t(m') = \left(\frac{M}{M'}\right)m' + \left(0.5 - \frac{M}{M'}\left(0.5\right)\right)$$

b) Derive the logic and computation to determine the closest pixel m to a location x in the original image. That is, given x, determine m where m $\in$[1,…,M]. You can assume

$$0.5 \le x \le M + 0.5 .$$

SOLUTION:

We can simply set $m = \text{round}(x)$. A special case is when $x = M + 0.5$ where we need to set $x = M$.

c) Derive the logic and computation to determine the two closest pixels m1 and m2 to a location x in the original image. Some special cases you might need to consider:

- x is integer valued
- x is less than 1.0
- x is greater than M
- x is equal to 1.0
- x is equal to M

SOLUTION:

```
If x is integer valued
    m1 = x
    m2 = x
Else
    If x < 1
            m1 = 1
            m2 = 2
    Else if x > m
            m1 = M-1
            m2 = M
    Else
            m1 = floor(x)                // largest integer smaller than x
            m2 = ceiling(x)              // smallest integer larger than x
    End
End
```

d) Now, think about the 2D case. Both dimensions will need to be mapped from the transformed image to the original image. Nearest neighbor interpolation still only requires the closest pixel. Bilinear interpolation now requires the four closest pixels.

Suppose mapping a pixel in the transformed image along the m dimension results in the locations m1 and m2 being the m-coordinates of the closest points, and that mapping a pixel in the transformed image along the n dimension results in the locations n1 and n2 being the n-coordinates of the closest points. List the coordinates of the four closest points in 2D. (This should be fairly straightforward but I want to get you thinking about the problem in 2D for your project).

SOLUTION:

The four closest pixels are simply

(m1, n1)
(m1, n2)
(m2, n1)
(m2, n2)

Homework #3
SOLUTION

1. A simple example shows that computing the median value of an image is not a linear operator. Let 1x3 image $F1 = [1, -2, 3]$, 1x3 image $F2 = [4,5,6]$, and $a = b = 1$. Let H be the median operator. We then have $H(F1 + F2) = median([5,3,9]) = 5$. We also have $H(F1) = median([1, -2, 3]) = 1$ and $H(F2) = median([4,5,6]) = 5$. Then, because $H(aF1 + bF2) \neq aH(F1) + bH(F2)$, the median operator is not linear.

2. In general, do affine transformations commute?

   SOLUTION
   Affine transformations, in general, do not commute. We just need to find a counterexample to prove this.

   Consider a rotation transformation and a shear transformation and their respective matrices:

   Rotation matrix = $T_R = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

   Shear matrix = $T_S = \begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

   Case 1: First rotate and then shear. That is, find where the pixel at (v, w) is mapped to by the combination of these transformations.

   First, the rotation:

$$[x' \quad y' \quad 1] = [v \quad w \quad 1] \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= [v\cos\theta - w\sin\theta \quad v\sin\theta + w\cos\theta \quad 1]$$

   Now, the shear:

$$[x \quad y \quad 1] = [x' \quad y' \quad 1] \begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= [v\cos\theta - w\sin\theta \quad v\sin\theta + w\cos\theta \quad 1] \begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= [v\cos\theta - w\sin\theta + s_v(v\sin\theta + w\cos\theta) \quad v\sin\theta + w\cos\theta \quad 1]$$

   So, $(x, y) = (v\cos\theta - w\sin\theta + s_v(v\sin\theta + w\cos\theta),\ v\sin\theta + w\cos\theta)$

   Case 2: First shear and then rotate. That is, find where the pixel at (v, w) is mapped to by the combination of these transformations.

First, the shear:

$$[x' \quad y' \quad 1] = [v \quad w \quad 1]\begin{bmatrix}1 & 0 & 0\\ s_v & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$= [v + s_v w \quad w \quad 1]$$

Now, the rotation:

$$[x \quad y \quad 1] = [x' \quad y' \quad 1]\begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$= [v + s_v w \quad w \quad 1]\begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$= [(v + s_v w)\cos\theta - w\sin\theta \quad (v + s_v w)\cos\theta - w\sin\theta \quad 1]$$

So, $(x, y) = \big((v + s_v w)\cos\theta - w\sin\theta\,,\ (v + s_v w)\cos\theta - w\sin\theta\big)$

This is not equal to (x, y) in case 1 above.

So, these two transformations do not commute and we have found our counterexample.

We can then conclude that affine transformations do not, in general, commute.

3. An affine transformation can be completely specified by its action on three points. What is the matrix for the transformation that maps the points A=(0,0) B=(1,0) and C=(0,1) to the points A'=(2,2) B'=(1,2) and C'=(2,1) respectively? (You can use Matlab.)

   Draw these points before and after the transformation. What is this transformation doing (in terms of scaling, rotating, shearing, translating, etc.)?

   SOLUTION
   Setup a system of equations representing the known mappings to solve for the six unknowns in the transformation matrix.

   Each mapping has the form

$$[x \quad y \quad 1] = [v \quad w \quad 1]\begin{bmatrix}t_{11} & t_{12} & 0\\ t_{21} & t_{22} & 0\\ t_{31} & t_{32} & 1\end{bmatrix}$$

   For example, the fact that $A = (0,0)$ is mapped to $A' = (2,2)$ gives

$$[2 \quad 2 \quad 1] = [0 \quad 0 \quad 1]\begin{bmatrix}t_{11} & t_{12} & 0\\ t_{21} & t_{22} & 0\\ t_{31} & t_{32} & 1\end{bmatrix}$$

   Now, the three mappings can be combined together in the following equation:

$$\begin{bmatrix}2 & 2 & 1\\1 & 2 & 1\\2 & 1 & 1\end{bmatrix} = \begin{bmatrix}0 & 0 & 1\\1 & 0 & 1\\0 & 1 & 1\end{bmatrix}\begin{bmatrix}t_{11} & t_{12} & 0\\t_{21} & t_{22} & 0\\t_{31} & t_{32} & 1\end{bmatrix}$$

Writing this as $G = FT$, we can solve for T as

$$T = F^{-1}G$$

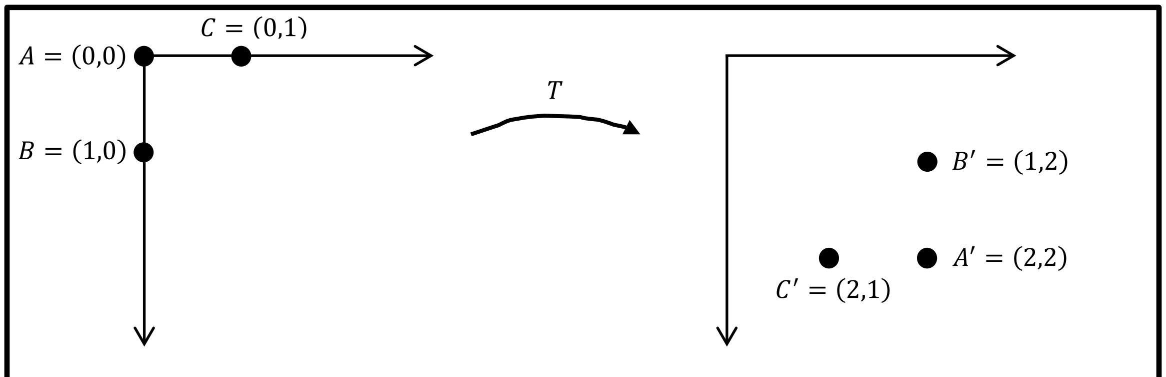
Using Matlab, we compute

$$F^{-1} = \begin{bmatrix}-1 & 1 & 0\\-1 & 0 & 1\\1 & 0 & 0\end{bmatrix}$$

and then we can solve for T

$$\boxed{T = \begin{bmatrix}-1 & 0 & 0\\0 & -1 & 0\\2 & 2 & 1\end{bmatrix}}$$

We now plot the points before and after the transformation:



We observe this transformation is a rotation by 180° followed by a translation by (2,2).

4. Suppose images $f(x,y)$ and $g(x,y)$ have histograms h_f and h_g. Suppose you form image $k(x,y)$ as the sum of these two images:

$$k(x,y) = f(x,y) + g(x,y).$$

(a) Under what conditions (on images f and g) can you determine the histogram h_k of image k in terms of the histograms h_f and h_g?

(That is, under what conditions can you determine h_k if you are only given h_f and h_g? Hint: this turns out to be fairly restrictive.)

(b) Explain how you would derive h_k in this case.

SOLUTION
The problem here is that you need to know the spatial correspondences between the intensity values in images f and g in order to determine the pixel intensities and counts in image k. Histograms do not convey information about the spatial locations of the pixel intensities.

One case in which you can determine h_k from h_f and h_g is when one or both of images f and g is <u>constant</u> (that is, has only one intensity value).

(b) Explain how you would derive h_k in this case.

SOLUTION
Without loss of generality, suppose image f is constant. That is,

$$f(x, y) = c$$

for some constant c.

Then, the number of pixels with value u_i in image k is the same as the number of pixels with value $u_i - c$ in image g.

Thus, the histogram of image k can be computed as

$$\boxed{h_k\ (u_i) = h_g\ (u_i - c)}$$

Note that this is the more general case of the situation where all the pixels in one of the images have value 0.

ALTERNATE SOLUTION

(a) Another case in which you can determine h_k from h_f and h_g is when images f and g are the same; i.e, $f(x, y) = g(x, y)$.

(b) Now, all the pixels in image k have twice the value of the corresponding pixels in image f (equivalently, image g). So, the number of pixels with value u_i in image k is the same as the number of pixels with value $\frac{u_i}{2}$ in image f.

Thus, the histogram of image k can be computed as

$$\boxed{h_k\ (u_i) = h_f\ \left(\frac{u_i}{2}\right)}$$

Homework #4
SOLUTION

1) Let *W* be the following 3x3 spatial filter

$W =$

1	1	1
0	0	0
-1	-1	-1

and let *F* be the following 6x6 image

$F =$

1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1

(a) Compute *G*, the result of applying the filter *W* to the image *F* using standard spatial filtering:

$$G(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} W(s,t) F(x+s, y+t)$$

G should have the same size *F*. Assume the zero-padding approach is used to deal with the case when the filter extends past the edge of the image.

SOLUTION

$G =$

-2	-3	-2	-1	0	0
0	0	0	0	0	0
2	3	1	-1	-3	-2
2	3	1	-1	-3	-2
0	0	0	0	0	0
0	0	1	2	3	2

(b) Give a brief description of what the spatial filter W does (in the general case, not just when applied to the image above).

SOLUTION

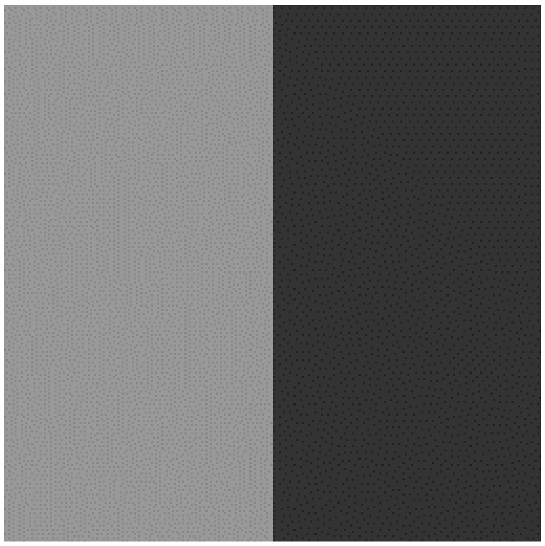
Detects large intensity changes in the vertical direction.

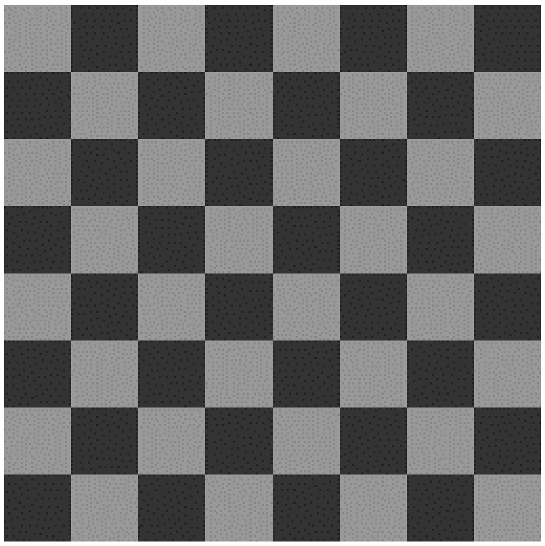
(c) Indicate how your result G above demonstrates this.

SOLUTION

The result in G has large (+ or -) values at the transition between 0 and 1 in the vertical direction.

2) Consider the two grayscale images below:


(a)


(b)

They are both 256x256 pixels in size. Image (a) has intensity value 150 in its left half and intensity value 50 in its right half. Image (b) is a checkerboard with squares measuring 32x32 pixels, half of intensity value 150 and the other half of intensity value 50.

(a) Do these two images have the same histogram? Briefly state why or why not.

SOLUTION

Yes, they have the same histogram. They have the number of pixels with intensity value 50 and the same number of pixels with intensity value 150 and they have no other pixels. Thus, their histograms will have only two non-zero values, the count of the number of pixels with intensity value 50 and the count of the number of pixels with intensity value 150. And, these two counts will be equivalent. (Not that both their unnormalized histograms and normalized histograms are equal.)

(b) Now, suppose you perform histogram equalization on the two images. Do the resulting images have the same histogram? Briefly state why or why not.

SOLUTION

Yes, the resulting images will have the same histogram. Since the transformation involved in histogram equalization depends only on an image's histogram, and since the original images have the same histogram, then the transformations for image (a) and image (b) will be equal. Histogram equalization maps a pixel's intensity value regardless of its spatial location. So the pixels with intensity value 50 will be mapped to the same intensity value in both images. The same is true for pixels with intensity value 150. Thus, there will be same number of pixels of a particular intensity in the transformed images and their histograms will again be the same.

(c) Suppose you blur each of the original images with a 3x3 averaging mask. Do the resulting images have the same histogram? Briefly state why or why not.

SOLUTION

No. To answer this, we focus on the boundaries between the regions of different intensity in the images. It is here that averaging mask will result in pixels with intensity values other than 50 or 150. Since the total boundary in image (a) is less than the total boundary in image (b) there will be fewer pixels with these other values in image (a). Thus the histograms of the blurred images will be different. Also note that the boundary in image (a) will result in pixels that have intensity value 83.3 or 116.7 (ignoring image boundary effects). However, the boundaries in image (b) will also result in pixels with intensity value 94.4 or 105.6.

3) In this problem, you will investigate a more efficient way to implement spatial filtering when all the filter coefficients have the same value. The motivation comes from the observation that as you slide the filter one pixel at a time over the image and compute the sum-of-products of image and filter values, you can use the results from the previous computation in the current computation.

Although the method can be generalized, we will consider the case in which all the filter coefficients have the value 1. And, we will also ignore the $1/n^2$ scaling factor that typically accompanies an averaging filter of size nxn.

SOLUTION 1

(a) Describe the algorithm you would use to compute the output value at location (x,y) given that you have already computed the result for location (x-1,y), for example, for an averaging filter of size nxn (think about what changes when you shift the filter by one pixel).

SOLUTION

Let G_{old} be the result already computed for location (x-1,y). To compute G_{new} at location (x,y) using G_{old}, we need to subtract the sum of pixels under the first column of the mask before it was moved (call this sum C_1) and add the sum of pixels under the last column of the mask after it was moved (call this sum C_n). So,

$$G_{new} = G_{old} - C_1 + C_n$$

(b) How many additions (in terms of n) does this require for each output pixel. Count subtractions as additions.

SOLUTION

We have already computed C_1. To compute C_n requires n-1 additions. Thus, to compute G_{new} from G_{old} requires **n+1** additions.

(c) Now, let's compare this with the standard approach of not using previous results. How many additions are required for each output pixel in this case. This should be in terms of n.

SOLUTION

The standard approach requires **n^2-1** additions for each output pixel.

(d) Compute the *computational advantage* of the more efficient approach. This is simply the ratio of the number of additions required by the standard approach to the number of additions required by the more efficient approach. Again, this should be in terms of n.

SOLUTION

The computational advantage is

$$\frac{n^2-1}{n+1} = \frac{(n-1)(n+1)}{n+1} = n-1$$

ALTERNATE SOLUTION

(a) Describe the algorithm you would use to compute the output value at location (x,y) given that you have already computed the result for location (x-1,y), for example, for an averaging filter of size nxn (think about what changes when you shift the filter by one pixel).

SOLUTION

Let G_{old} be the result already computed for location (x-1,y). To compute G_{new} at location (x,y) using G_{old}, we need to subtract the sum of pixels under the first column of the mask before it was moved (call this sum C_1) and add the sum of pixels under the last column of the mask after it was moved (call this sum C_n). So,

$$G_{new} = G_{old} - C_1 + C_n$$

(b) How many additions (in terms of n) does this require for each output pixel. Count subtractions as additions.

SOLUTION

It takes n-1 additions to compute C_1. To compute C_n also requires n-1 additions. Thus, to compute G_{new} from G_{old} requires **2n** additions.

(c) Now, let's compare this with the standard approach of not using previous results. How many additions are required for each output pixel in this case. This should be in terms of n.

SOLUTION

The standard approach requires **n^2-1** additions for each output pixel.

(d) Compute the *computational advantage* of the more efficient approach. This is simply the ratio of the number of additions required by the standard approach to the number of additions required by the more efficient approach. Again, this should be in terms of n.

SOLUTION

The computational advantage is

$$\frac{n^2 - 1}{2n}$$

Homework #5
SOLUTION

1. Problem 10.22 in the text.

(a) The problem here is to transform the slope-intercept form of a line

$$y = ax + b$$

into the normal form

$$x\cos\theta + y\sin\theta = \rho .$$

That is, we need to express θ and ρ in terms of *a* and *b*. An easy way to do this is to start with the normal form and work backwards. Express the normal form as

$$y = -(\cot\theta)x + \rho/\sin\theta .$$

Equating terms with the slope-intercept form, we get

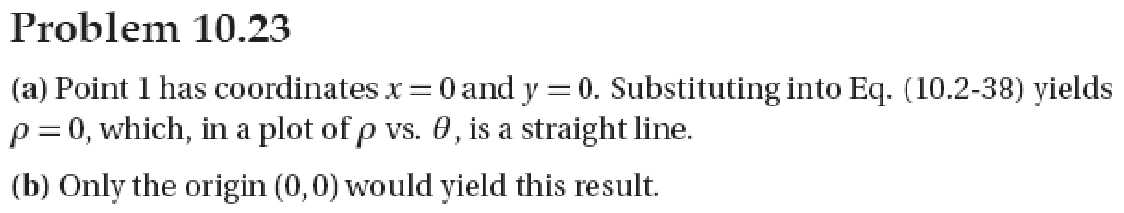
$$a = -(\cot\theta) \text{ and } b = \rho/\sin\theta .$$

Thus, the solution is

$$\theta = arc\cot(-a) \text{ and } \rho = b\sin\theta$$

(b) $\theta = arc\cot(2) = 26.6°$ and $\rho = (1)\sin\theta = 0.45$.

2. Problem 10.23 (a) and (b). (You don't need to do part (c)).



Problem 10.23

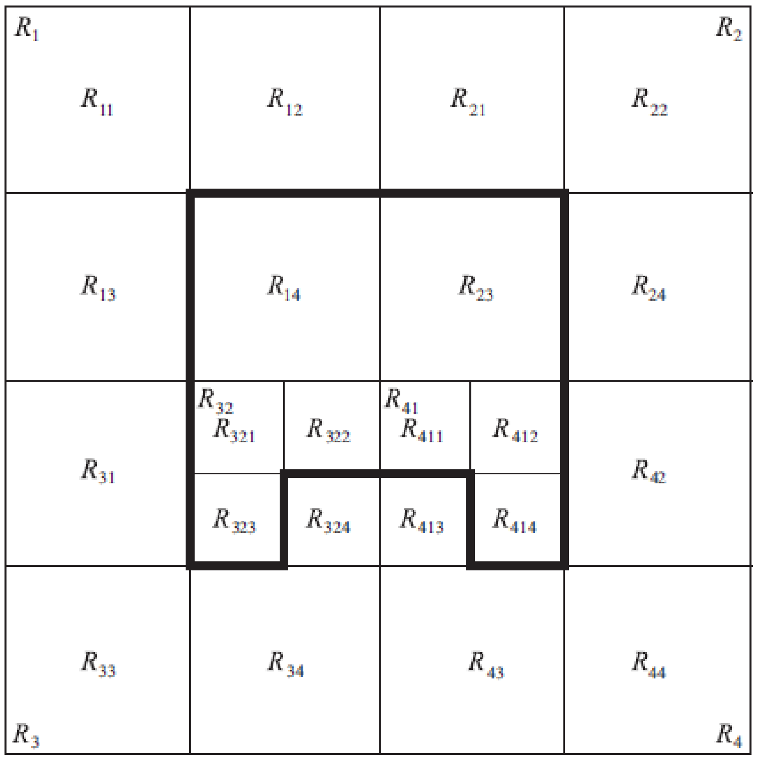
(a) Point 1 has coordinates $x = 0$ and $y = 0$. Substituting into Eq. (10.2-38) yields $\rho = 0$, which, in a plot of ρ vs. θ, is a straight line.

(b) Only the origin $(0, 0)$ would yield this result.

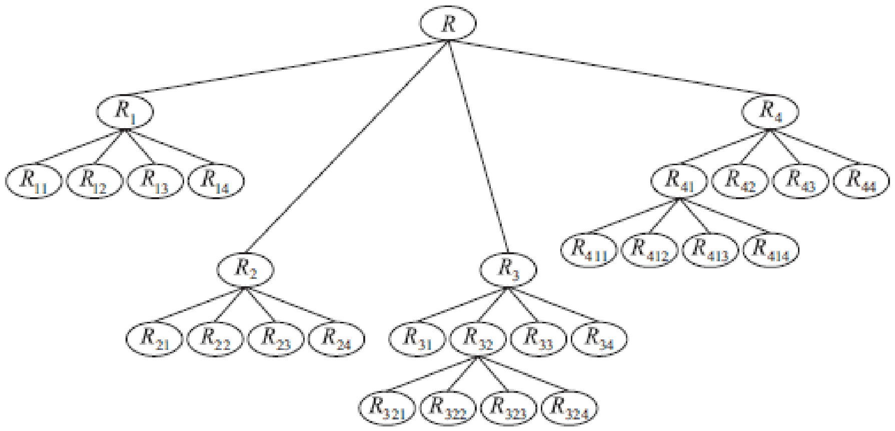
3. Problem 10.39 in the text.

Problem 10.39

The region splitting is shown in Fig. P10.39(a). The corresponding quadtree is shown in Fig. P10.39(b).



(a)



(b)

Figure P10.39

4. Problem 10.43 in the text.

Problem 10.43

The first step in the application of the watershed segmentation algorithm is to build a dam of height max + 1 to prevent the rising water from running off the ends of the function, as shown in Fig. P10.43(b). For an image function we would build a box of height max + 1 around its border. The algorithm is initialized by setting $C[1] = T[1]$. In this case, $T[1] = \{g(2)\}$, as shown in Fig. P10.43(c) (note the water level). There is only one connected component in this case: $Q[1] = \{q_1\} = \{g(2)\}$.

Next, we let $n = 2$ and, as shown in Fig. P10.43(d), $T[2] = \{g(2), g(14)\}$ and $Q[2] = \{q_1; q_2\}$, where, for clarity, different connected components are separated by semicolons. We start construction of $C[2]$ by considering each connected component in $Q[2]$. When $q = q_1$, the term $q \cap C[1]$ is equal to $\{g(2)\}$, so condition 2 is satisfied and, therefore, $C[2] = \{g(2)\}$. When $q = q_2$, $q \cap C[1] = \emptyset$ (the empty set) so condition 1 is satisfied and we incorporate q in $C[2]$, which then becomes $C[2] = \{g(2); g(14)\}$ where, as above, different connected components are separated by semicolons.

When $n = 3$ [Fig. P10.43(e)], $T[3] = \{2, 3, 10, 11, 13, 14\}$ and $Q[3] = \{q_1; q_2; q_3\} = \{2, 3; 10, 11; 13, 14\}$ where, in order to simplify the notation we let k denote $g(k)$. Proceeding as above, $q_1 \cap C[2] = \{2\}$ satisfies condition 2, so q_1 is incorporated into the new set to yield $C[3] = \{2, 3; 14\}$. Similarly, $q_2 \cap C[2] = \emptyset$ satisfies condition 1 and $C[3] = \{2, 3; 10, 11; 14\}$. Finally, $q_3 \cap C[2] = \{14\}$ satisfies condition 2 and $C[3] = \{2, 3; 10, 11; 13, 14\}$. It is easily verified that $C[4] = C[3] = \{2, 3; 10, 11; 13, 14\}$.

When $n = 5$ [Fig. P10.43(f)], we have, $T[5] = \{2, 3, 5, 6, 10, 11, 12, 13, 14\}$ and $Q[5] = \{q_1; q_2; q_3\} = \{2, 3; 5, 6; 10, 11, 12, 13, 14\}$ (note the merging of two previously distinct connected components). Is is easily verified that $q_1 \cap C[4]$ satisfies condition 2 and that $q_2 \cap C[4]$ satisfied condition 1. Proceeding with these two connected components exactly as above yields $C[5] = \{2, 3; 5, 6; 10, 11; 13, 14\}$ up to this point. Things get more interesting when we consider q_3. Now, $q_3 \cap C[4] = \{10, 11; 13, 14\}$ which, becuase it contains two connected components of $C[4]$, satisfies condition 3. As mentioned previously, this is an indication that water from two different basins has merged and a dam must be built to prevent this condition. Dam building is nothing more than separating q_3 into the two original connected components. In this particular case, this is accomplished by the dam shown in Fig. P10.43(g), so that now $q_3 = \{q_{31}; q_{32}\} = \{10, 11; 13, 14\}$. Then, $q_{31} \cap C[4]$ and $q_{32} \cap C[4]$ each satisfy condition 2 and we have the final result for $n = 5$, $C[5] = \{2, 3; 5, 6; 10, 11; 13; 14\}$.

Continuing in the manner just explained yields the final segmentation result shown in Fig. P10.43(h), where the "edges" are visible (from the top) just above the water line. A final post-processing step would remove the outer dam walls to yield the inner edges of interest.

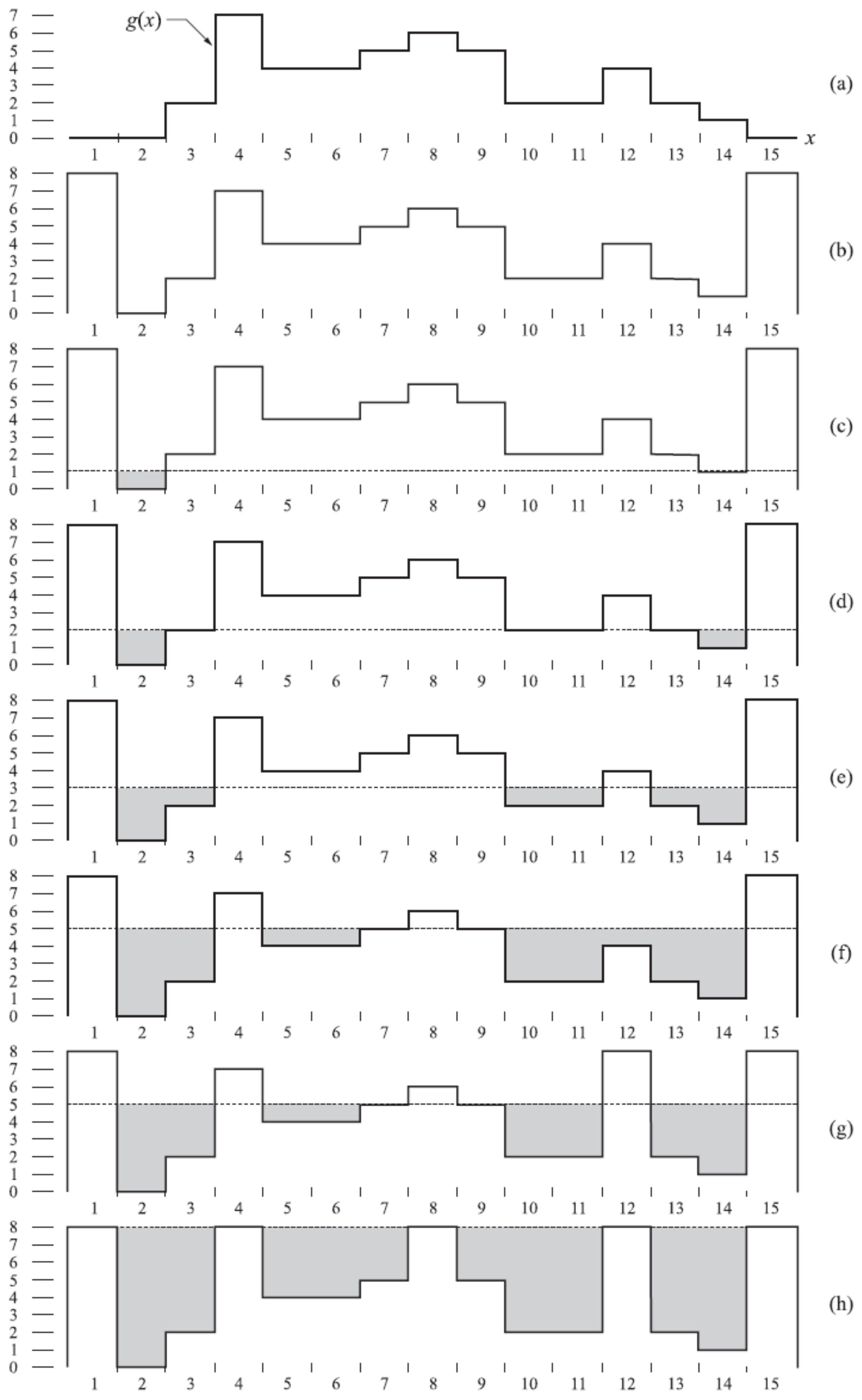


Figure P10.43