

SIMG-782 Introduction to Digital Image Processing

Homework 3 – Due September 29, 2005

1. A binary array that represents a portion of a black and white image is given below. Perform the operations listed below on this piece of image. Assume that all of the pixels that surround this segment contain black background.

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

- (a) Dilation with the structuring element

①	1	1
---	---	---

 The origin is shown with a circled element.

Generate the image array

```

1  <hw3prob1solutions.pro 1>≡
    ;SIMG-782 FALL 2004
    ;HOMEWORK 3 SOLUTIONS
    PRO hw3P1
    ;PROBLEM 1: First construct the number array that represents the image.
    A=0,0,0,0,0,0,0],[0,0,1,1,0,0,0],[0,0,0,1,0,0,0],[0,0,0,1,1,0,0],$

    [0,0,1,1,1,1,0],[0,0,1,1,1,0,0],[0,1,0,1,0,1,0],[0,0,0,0,0,0,0

    B1=[1,1,1]
    C1=Dilate(byte(A),byte(B1),0)
    print,format='(Structuring Element B1=[" ,3i1,"])',B1
    print,'C1=dilate(A,B1,0)'
    print,' A C1'
    print,[A,C1],format='(7i1,1x,7i1)'

```

This code is written to file hw3prob1solutions.pro.

This produces the following output. Note that the formatting somewhat reduces the amount of space required. We use the “0” argument in dilate to specify the origin location. Note that IDL does gives the wrong value in position $C_1[6,6]$, which should be a 1.

```
Structuring Element B1=[111]
C1=dilate(A,B1,0)
      A      C1
0000000 0000000
0011000 0011110
0001000 0001110
0001100 0001111
0011110 0011111
0011100 0011111
0101010 0111110
0000000 0000000
```

- (b) Erosion with the structuring element

Ⓐ	1	1
---	---	---

 This can be done with the following code:

2 $\langle hw3prob1solutions.pro\ 1 \rangle + \equiv$ $\triangleleft 1\ 3 \triangleright$

```
C2=erode(A,B1,0)
print,'C2=erode(A,B1,0)'
print,'      A      C2'
print,[A,C2],format='(7i1,1x,7i1)'
```

```
C2=erode(A,B1,0)
```

A	C2
0000000	0000000
0011000	0000000
0001000	0000000
0001100	0000000
0011110	0011000
0011100	0010000
0101010	0000000
0000000	0000000

(c) Dilation with the structuring element

1	①
0	1

```
3  <hw3prob1solutions.pro 1>+≡
    B2=1,1],[0,1
    C3=dilate(A,B2,1,0)
    print
    print,format='(Structuring Element="//(2i1))',B2
    print
    print,'C3=dilate(A,B2,1,0)
    print,'  A      C3'
    print,[A,C3],format='(7i1,1x,7i1)'
```

<2 4>

The result is
Structuring Element=
11
01

C3=dilate(A,B2,1,0)

A	C3
0000000	0000000
0011000	0111000
0001000	0011000
0001100	0011100
0011110	0111110
0011100	0111110
0101010	1111110
0000000	0101010

(d) Erosion with the structuring element

1	Ⓢ
0	1

4 *hw3prob1solutions.pro* 1)+≡

◀3 5▶

```

C4=erode(A,B2,1,0)
print,'C4=erode(A,B2,1,0)'
print,'    A        C4'
print,[A,C4],format='(7i1,1x,7i1)'

```

The result is

```
C4=erode(A,B2,1,0)
```

A	C4
0000000	0000000
0011000	0001000
0001000	0000000
0001100	0000100
0011110	0001100
0011100	0001000
0101010	0000000
0000000	0000000

(e) Opening with each of the above structuring elements.

5 *<hw3prob1solutions.pro 1>+≡* <4 6>

```
C5=dilate(erode(A,B1,0),B1,0)
print
print,'Opening with B1 c5=dilate(erode(A,B1,0),B1,0)'
print,'  A      C2      C5'
print,format='(7I1," ",7I1," ",7i1)',[A,C2,C5]

C6=dilate(erode(A,B2,1,0),B2,1,0)
print
print,'Opening with B2 c6=dilate(erode(A,B2,1,0),B2,1,0)'
print,'  A      C4      C6'
print,format='(7I1," ",7I1," ",7i1)',[A,C4,C6]
```

Opening with B1 c5=dilate(erode(A,B1,0),B1,0)

A	C2	C5
0000000	0000000	0000000
0011000	0000000	0000000
0001000	0000000	0000000
0001100	0000000	0000000
0011110	0011000	0011110
0011100	0010000	0011100
0101010	0000000	0000000
0000000	0000000	0000000

Opening with B2 c6=dilate(erode(A,B2,1,0),B2,1,0)

A	C4	C6
0000000	0000000	0000000
0011000	0001000	0011000
0001000	0000000	0001000
0001100	0000100	0001100
0011110	0001100	0011100
0011100	0001000	0011100
0101010	0000000	0001000
0000000	0000000	0000000

(f) Closing with each of the above structuring elements.

6 $\langle hw3prob1solutions.pro \ 1 \rangle + \equiv$ <5

```

C7=erode(dilate(A,B1,0),B1,0)
print
print,'Closing with B1 C7=erode(dilate(A,B1,0),B1,0)'
print,'  A          C1          C7'
print,format='(7I1," ",7I1," ",7i1)',[A,C1,C7]

C8=erode(dilate(A,B2,1,0),B2,1,0)
print
print,'Closing with B2 C8=erode(dilate(A,B2,1,0),B2,1,0)'
print,'  A          C3          C8'
print,format='(7I1," ",7I1," ",7i1)',[A,C3,C8]

END

```

Closing with B1 C7=erode(dilate(A,B1,0),B1,0)

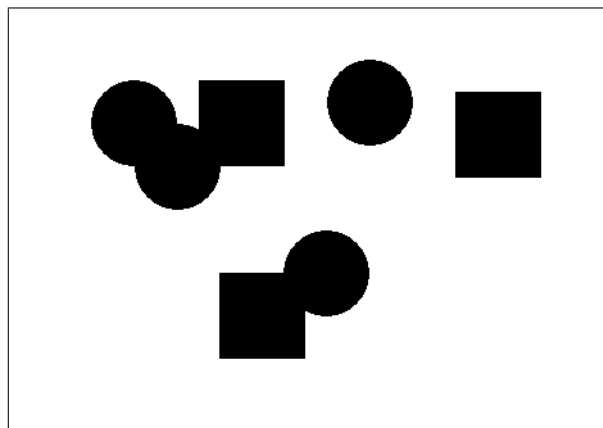
A	C1	C7
0000000	0000000	0000000
0011000	0011110	0011000
0001000	0001110	0001000
0001100	0001111	0001100
0011110	0011111	0011100
0011100	0011111	0011100
0101010	0111110	0111000
0000000	0000000	0000000

Closing with B2 C8=erode(dilate(A,B2,1,0),B2,1,0)

A	C3	C8
0000000	0000000	0000000
0011000	0111000	0011000
0001000	0011000	0001000
0001100	0011100	0001100
0011110	0111110	0011110
0011100	0111110	0011110
0101010	1111110	0101010
0000000	0101010	0000000

Note: IDL provides functions `morph_open` and `morph_close` that do opening and closing.

2. Shown below is a binary image with some disks and squares, some of which overlap. This is the image BW3.png in the images directory. For each question provide an algorithm that uses morphological and logical operations to answer the questions. The answers may be in the form of pseudocode with a block diagram. For extra credit, you may submit an IDL program that implements the algorithms. Assume all disks are the same size and all squares are the same size.



Binary image BW3.png with touching objects

- (a) Develop a technique based on morphology and connected components to count the number of disks in the above image.
- (b) Develop a technique based on morphology and logic to count the boundary pixels in the above image.
- (c) Develop a technique based on morphology and logic to count the pixels that are common to two or more objects.

Solution Counting disks is difficult because a disk would fit within a square so that erosion with a disk model would not work. Similarly, the overlap of disks and squares make it difficult to use hit-or-miss on disks. An easier approach is to remove the squares and look for disks in what remains. The following code chunk does that. Since the square and the disk each appear in isolation it is easy to make a model for each simply by selecting them out.

Once the squares are selected, we can build an image that has just squares in it by doing an opening with the square structuring element. Since we have already eroded, we can do that just by dilation.

Erosion with the disk structuring element leads to hits in both the disks and the squares. We determine which hits correspond to the squares and remove them. We can then construct a disk image by dilating the remaining points with the disk structuring element.

It is helpful in identifying pixels in the overlap area to have the individual disk images in separate frames. We can use a 3D array to stack the individual images.

Finally, finding the overlap is aided by adding the various image arrays. This leads to pixels with different values in the overlap than in the original.

The boundary is found by a dilation and masking operation in the last step of the program.

The results are shown following the program listing.

```
8  <hw3prob2solutions.pro 8>≡
    pro hw3P2,imgpath
    ;Homework 3 problem 2 2004
    ;Open the file
    closeall
    loadct,0
    A=read_image(imgpath+'BW3.png')
    sa=size(A,/dim)
    window,0,xs=sa[0],ys=sa[1],title='Original',xpos=0,ypos=0
    TV,A
    ;Work with the complement since objects in A are at level 0
    Ac=A NE 255

    ;We need to find the square and the circle in isolation so
    ;we can build the corresponding structuring elements. First
    ;find all the connected components.
    r=Label_Region(Ac)
    tek_color
```



```

window,1,xs=sa[0],ys=sa[1],title='Components',xpos=sa[0]+5,ypos=0
TV,r

hr=histogram(r)
ir=where(hr > 0)
print,hr[ir]

;By inspection we see that the square must be object 3 and the
;disk object 4. They are the two smallest objects, and we know
;the disk has fewer pixels than the square.

;Display the square in isolation (It is fortunate that the image
;has just one such, or we would have to refine the selection.)
i3=where(r eq 3)
B=r eq 3
window,2,xs=sa[0],ys=sa[1],title='Object 3',xpos=0,ypos=sa[1]+25
TV,B

;To build the square structuring element, find the bounding box
x3=i3 mod sa[0]
y3=i3/sa[0]
print,"Left and right boundaries of square: ",[min(x3),max(x3)]
print,"Top and bottom boundaries of square: ",[min(y3),max(y3)]
;Here is the structuring element
B0=B[min(x3):max(x3),min(y3):max(y3)]

;Now erode Ac to find the locations of squares
B1=Erode(Ac,B0)
window,3,xs=sa[0],ys=sa[1],title='Square Locations',xpos=0,ypos=2*(sa[1]+25)
TV,B1

;Dilate to construct an image of the squares
B2=dilate(B1,B0)

;Form a unique label for each square
rsq=label_region(B2)
window,4,xs=sa[0],ys=sa[1],xpos=sa[0]+5,ypos=2*(sa[1]+25),title='Labeled Squares'
TV,rsq
nsquares=max(rsq) ;Number of squares in the image

;Find the isolated disk and build a structuring element
i4=where(r eq 4)
C=r eq 4
x4=i4 mod sa[0]
y4=i4/sa[0]

```

```

print,"Left and right boundaries of the disk: ",[min(x4),max(x4)]
print,"Top and bottom boundaries of the disk: ",[min(y4),max(y4)]
window,5,xs=sa[0],ys=sa[1],title='Object 4',xpos=sa[0]+5,ypos=sa[1]+25
tv,C

;Disk structuring element
C0=C[min(x4):max(x4),min(y4):max(y4)]

;Find all the places that can contain the element C0. This will include
;the squares, since the disk fits within a square.
C1=erode(Ac,C0)
window,6,xs=sa[0],ys=sa[1],title='Disk Erosion',xpos=0,ypos=0.5*(sa[1]+25)
TV,C1

;Now remove the locations that fall within the squares.
C3=C1 AND NOT B2
window,7,xs=sa[0],ys=sa[1],title='Disk Unique Locations',xpos=sa[0]+5,ypos=0.5*(sa[1]+25)
TV,C3

;Dilate each location separately so that we can isolate the
;overlapping disks.
rd=label_region(C3)
hd=histogram(rd)
id=where(hd GT 0)
print,id
print,hd[id]
;Number of disks. Count all but the background.
ndisk=n_elements(id)-1

;Construct an array to hold the results. Here a stack is useful.
;Each layer will hold a unique disk
C=bytarr(ndisk,sa[0],sa[1])
for k=1,ndisk do C[k-1,*]=dilate((rd eq id[k]),C0)*(nsquares+k)

window,8,xs=sa[0],ys=sa[1],title='Separated Disks',xpos=0,ypos=(sa[1]+25)
CT=total(C,1)
TV,CT

window,9,xs=sa[0],ys=sa[1],title='All Objects',xpos=sa[0]+5,ypos=(sa[1]+25)
TV,CT+rsq

;Find the common pixels by adding up all of the object images
CP=RSQ GT 0
CP=CP + total(C GT 0,1)
window,10,xs=sa[0],ys=sa[1],title='Common Pixels',xpos=2*(sa[0]+5),ypos=1.5*(sa[1]+25)

```

tv,cp

;Find the number of pixels in the intersection

print,format='("Number of common pixels=",I5)',total(cp eq 2)

;Find the boundary pixels

S=replicate(1,3,3)

;There are a number of ways to do this. One of them is

BP=(Erode(Ac,S)EQ 0) AND Ac

window,11,xs=sa[0],ys=sa[1],title='Boundary Pixels',xpos=2*(sa[0]+5),ypos=2*(sa[1]+25)

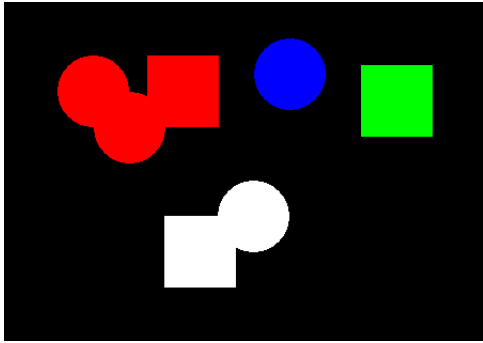
TV,BP

;Of course, since we have located the individual objects, we could also find

;the individual boundaries.

end

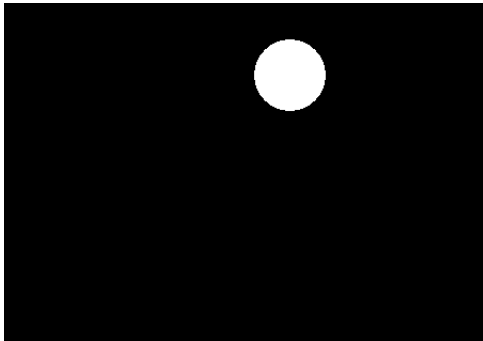
This code is written to file hw3prob2solutions.pro.



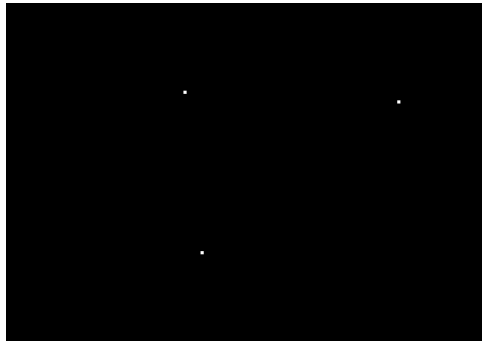
Labeled Components



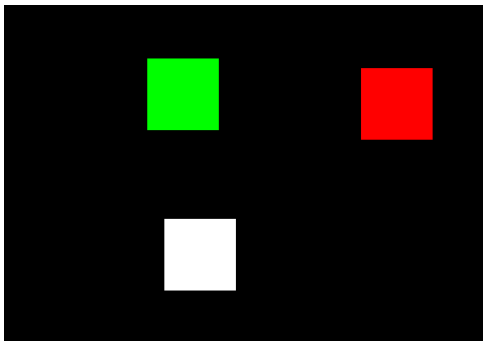
Isolated Square



Isolated Disk



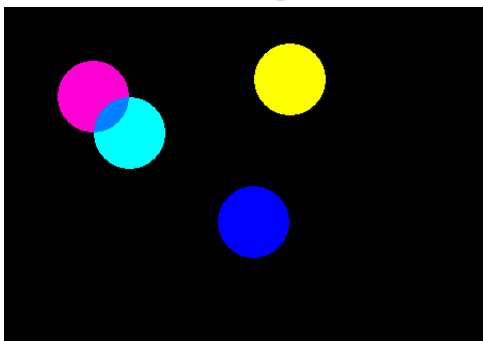
Result of erosion with square



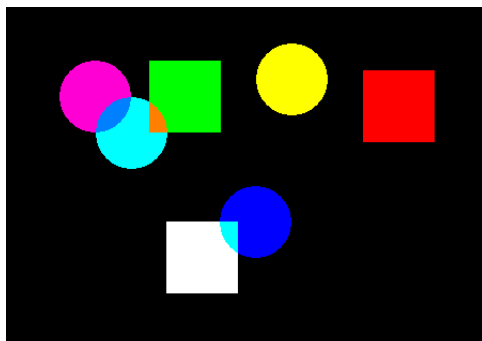
Labeled Squares



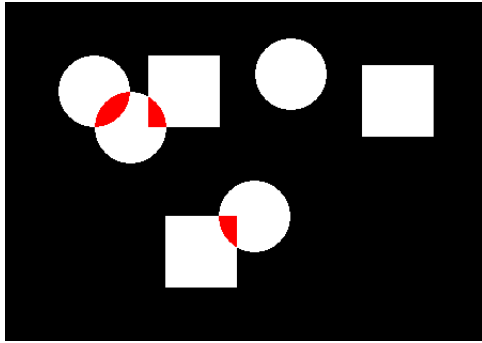
Result of erosion with disk



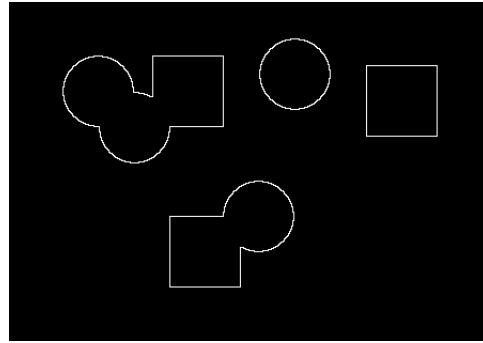
Separated Disks



All Objects



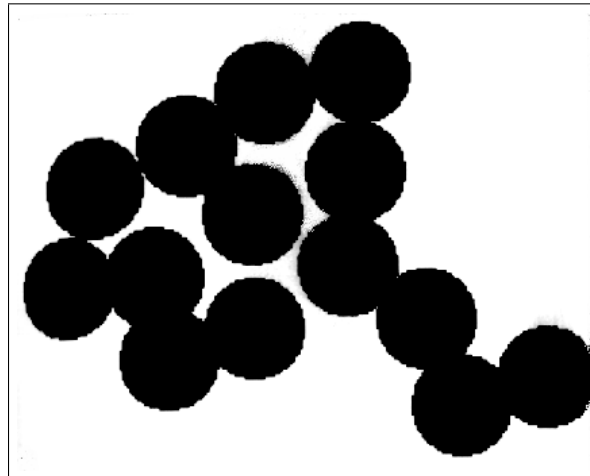
Common Pixels



Boundary

The number of common pixels is found by counting the number of red pixels, and is 1286.

3. Shown below is an image with a number of touching black disks against a white background. Construct a program that will count the number of disks. [Hint: The program `disc.pro` in the programs directory can be used to help in constructing disk-shaped structuring elements of different sizes.]

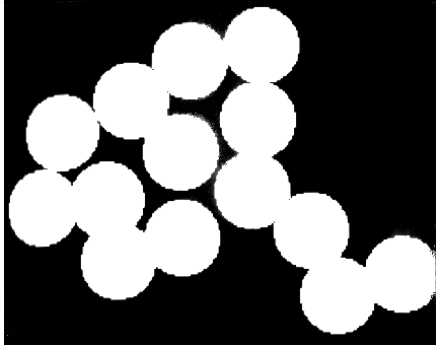


Binary image BWFatCircles.png showing a number of touching disks.

Solution: The circles can be counted by eroding with a structuring element that is a disk about as big as those in the image (a little smaller is better because too big will lose the objects). After erosion, the separate components can be counted by using `Label_Region`.

```
A=Read_Image('BWFatCircles.png')
B=255*(NOT A) ;Makes the background 0
S=disc(70) LE 35 ;Disk structuring element of radius 35
C=erode(B,S)
D=label_region(D,/ALL)
h=histogram(D) ;To count the components
k=where(h > 0) ;Only want the bins that have counts
n=n_elements(k)-1 ;One of the bins is the background
print,'Number of disks = ',n
```

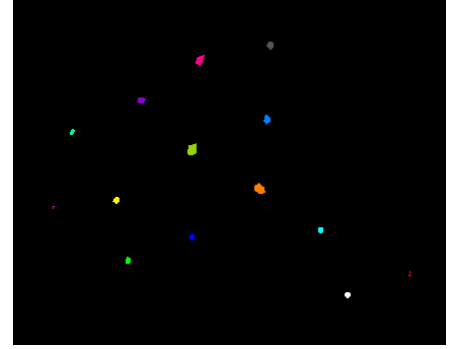
The result is 14 objects. Images B, C and D are shown below.



Negative Image

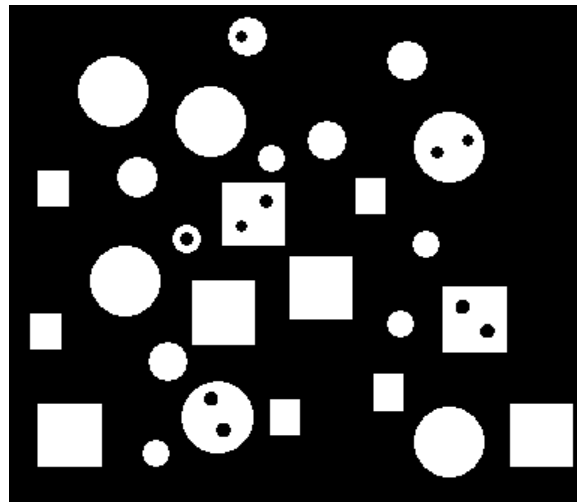


Eroded Image



Labeled Image

4. Shown below is an image with circles and squares of various sizes. Some of the objects have one or two holes in them. You are to provide an algorithm that uses morphological and logical operations to answer the questions below. The answers may be in the form of pseudocode with a block diagram. For extra credit, you may submit an IDL program that implements the algorithms. The image is in the images folder with the name 'blocks1.png'



Binary image blocks1.png showing a variety of circles and squares, some of which have one or two holes.

- (a) What fraction of the image pixels are white? We simply count the number of nonzero pixels as a fraction of the total number of pixels in the image. Since the **TOTAL** function always returns a floating point value, we do not need to worry about integer division.
- ```

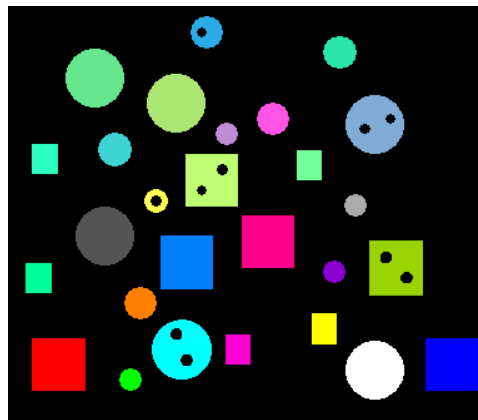
Pro hw3_1,A
fname='\blocks1.png'
A=read_image(fname)
Print,total(A GT 0)/n_elements(A)
END
RESULT: 21.3% white pixels

```
- (b) How many objects are in the image? Here we can just use **LABEL\_REGION** and count the number of colors other than color 0. We do the counting by using the histogram function. It would also be OK to use **nobjects=MAX(LA)** because we know that **LABEL\_REGION** numbers consecutively.

```

PRO hw3_2,A,LA,nobjects
LA=Label_Region(A)
ha=histogram(LA)
ia=where(ha gt 0)
nobjects=n_elements(ia)-1
print,'Number of objects=',nobjects
sa=size(A,/dim)
window,/free,xsize=sa[0],ysize=sa[1]
;Remember the old color palette
tv1ct,rr,gg,bb,/get
;Change to the TEK color palette
tek_color
TV,LA
;Reset the color palette
tv1ct,rr,gg,bb
END

```



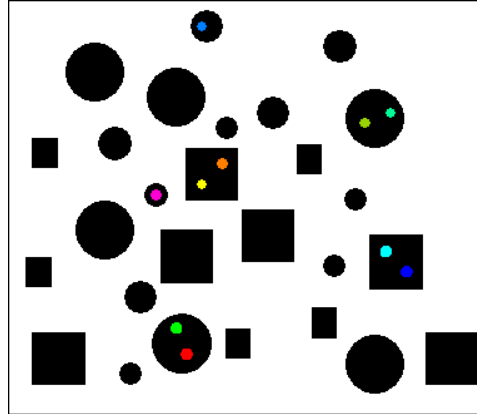
Number of objects= 27

- (c) How many holes are in the image? To count the holes, we construct  $A^c$  and count the objects in it. We then subtract one for the background.

```

PRO hw3_3,A,LC
Ac=A EQ 0
LC=label_region(Ac,/ALL)
hc=histogram(LC)
ic=where(hc gt 0)
;The number of holes does not count the background.
;Also allow for 0-based indexing.
print,'Number of Holes=',n_elements(ic)-2
sa=size(A,/dim)
window,/free,xsize=sa[0],ysize=sa[1]
tv1ct,rr,gg,bb,/get
tek_color
TV,LC
tv1ct,rr,gg,bb
END

```



Number of Holes= 10

(d) How many objects have one or more holes?

```

PRO hw3.4,A,LA,LC,count
;Find all the holes
Ac=A EQ 0
LC=label_region(Ac,/ALL)
;Construct an array with the holes filled in
Afill=(A GT 0) OR (LC GT 1)
;Display the arrays
sa=size(A,/dim)
window,/free,xsize=sa[0],ysize=sa[1]
tv!ct,rr,gg,bb,/get
tek_color
TV,Afill
window,/free,xsize=sa[0],ysize=sa[1]
TV,LC
;Count the objects with holes. First we
;find all the objects and then match up
;the object labels and the hole labels.
LA=label_region(Afill)
window,/free,xsize=sa[0],ysize=sa[1]
TV,LA
ha=histogram(LA)
ia=where(ha ge 0)
print,format='("Objects",3x,"Count",/, (i2,5x,i7))',$
 [transpose(ia),transpose(ha[ia])]
;Each element of ia corresponds to a filled
;object. Object k is labeled ia[k]. For each
;object that is not background,
;determine whether it has holes.
c=0
print
print,"k ia[k] N C"
For k=1,n_elements(ia)-1 DO BEGIN
 B=bytarr(sa[0],sa[1]); Make an array with one object
 ik=Where(LA eq ia[k]); ;Fill in the object
 IF MIN(ik) GE 0 THEN B[ik]=1
 ;Now see if any of the object pixels match the
 ;hole image LC. Counts if one or more holes.
 IF MAX(B AND (LC GT 0)) GT 0 THEN c++
 print,[k,ia[k],n_elements(ik),c],format='(I2,1x,I2,3x,I5,2x,I1)'
END
Print,'Number of objects with one or more holes=',count

```



```

tv1ct,rr,gg,bb
END
IDL> hw3_4,A,LA,LC,count

```

```

Objects Count

```

```

0 99502
1 1725
2 1806
3 248
4 1764
5 1795
6 480
7 500
8 516
9 1892
10 504
11 1806
12 248
13 1764
14 1725
15 248
16 277
17 1764
18 480
19 504
20 559
21 1725
22 248
23 516
24 1725
25 1725
26 542
27 516

```

```

k ia[k] N C
1 1 1725 00
2 2 1806 00
3 3 248 00
4 4 1764 00
5 5 1795 11
6 6 480 11
7 7 500 11
8 8 516 11
9 9 1892 22
10 10 504 22
11 11 1806 22
12 12 248 22
13 13 1764 22
14 14 1725 22
15 15 248 22
16 16 277 33
17 17 1764 44
18 18 480 44
19 19 504 44
20 20 559 44
21 21 1725 55
22 22 248 55
23 23 516 55
24 24 1725 55
25 25 1725 55
26 26 542 55
27 27 516 66

```

```

Number of objects with one or more holes= 6

```

- (e) How many square objects are in the image? We can distinguish squares from disks by having a square corner. Looking for a square corner enables us to find squares of any size, while using a full-size hit-or-miss mask for each square would be more complicated.

```

PRO hw3_5,A,B,nsquares
;Use a hit-or-miss to do the count
M={\Tt{ }1,1,1],[1,0,0],[1,0,0\nwendquote}
H={\Tt{ }0,0,0],[0,1,0],[0,0,0\nwendquote}
B=morph_hitormiss(A,H,M)
nsquares=Fix(Total(B))

```

```
print,'Number of square objects=',nsquares
END
```



Number of square objects= 11

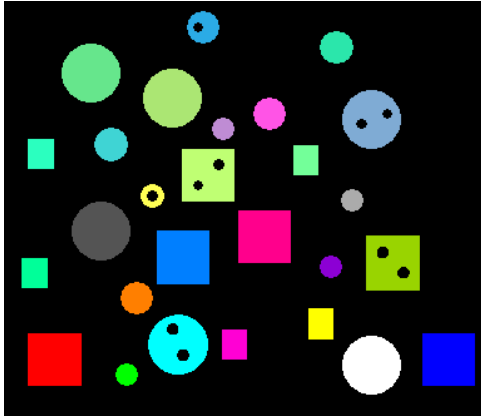
(f) Identify the square objects that have holes.

```
PRO hw3_6,A,Q,LA,LC,C,Count
;Find the square objects that have holes.
;
tvlct,rr,gg,bb,/get
tek_color
sa=size(A,/dim)
;Find the square objects
M={\Tt{1,1,1],[1,0,0],[1,0,0\nwendquote}
H={\Tt{0,0,0],[0,1,0],[0,0,0\nwendquote}
Q=morph_hitormiss(A,H,M)
window,/free,xsize=sa[0],ysize=sa[1],Title='Corners of Squares'
TV,Q
;Find the objects with holes
Ac=A EQ 0
LC=label_region(Ac,/ALL)
Afill=(A GT 0) OR (LC GT 1)
window,/free,xsize=sa[0],ysize=sa[1],title='Array Afill'
TV,Afill
window,/free,xsize=sa[0],ysize=sa[1],title='Array LC'
TV,LC
LA=label_region(Afill)
window,/free,xsize=sa[0],ysize=sa[1],title='Array LA'
TV,LA
ha=histogram(LA)
ia=where(ha ge 0)
count=0
C=bytarr(sa[0],sa[1])
For k=1,n_elements(ia)-1 DO BEGIN
 B=bytarr(sa[0],sa[1])
 ik=Where(LA eq ia[k])
 IF MIN(ik) GE 0 THEN B[ik]=1
 IF MAX(B AND (LC GT 0)) GT 0 THEN C[ik]=ia[k]
END
window,/free,xsize=sa[0],ysize=sa[1],title='Elements with Holes'
;Dilation is to emphasize corners of squares on the display
TV,C + dilate(Q,replicate(1,5,5))
;Match them up
```

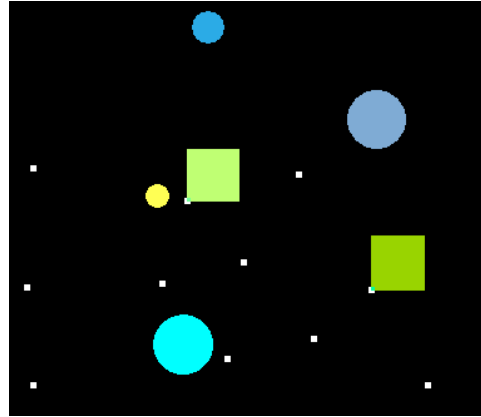
```

;We know that each Q marker has one pixel, so we can
;just count pixels in the intersection of Q with C.
Print,'Number of squares with holes=',total(Q AND C)
tv!ct,rr,gg,bb
end

```



Labeled Objects



Two of the objects with holes are marked as squares

(g) Identify the circular objects that have no holes.

```

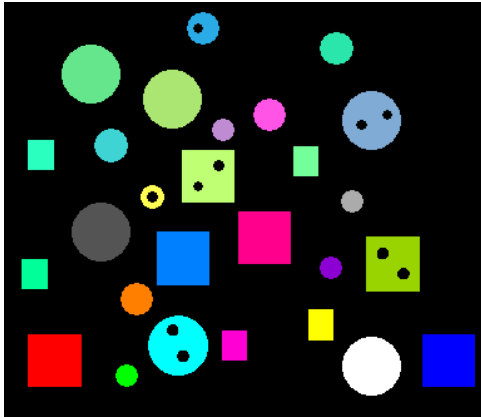
PRO hw3_7,A,Q,LA,LC,C,nobjects,nsquares
;Find the square objects that have holes.
;
tv!ct,rr,gg,bb,/get
tek_color
sa=size(A,/dim)
;Find the square objects
M={\Tt{1,1,1],[1,0,0],[1,0,0\nwendquote}
H={\Tt{0,0,0],[0,1,0],[0,0,0\nwendquote}
Q=morph_hitormiss(A,H,M)
window,/free,xsize=sa[0],ysize=sa[1],Title='Corners of Squares'
TV,Q
;Find the objects with holes
Ac=A EQ 0
LC=label_region(Ac,/ALL)
Afill=(A GT 0) OR (LC GT 1)
window,/free,xsize=sa[0],ysize=sa[1],title='Array Afill'
TV,Afill
window,/free,xsize=sa[0],ysize=sa[1],title='Array LC'
TV,LC
LA=label_region(Afill) ;Counts all objects & has holes filled
window,/free,xsize=sa[0],ysize=sa[1],title='Array LA'
TV,LA
ha=histogram(LA)
ia=where(ha ge 0)
count=0
C=bytarr(sa[0],sa[1])
For k=1,n_elements(ia)-1 DO BEGIN
 B=bytarr(sa[0],sa[1])
 ik=Where(LA eq ia[k])
 IF MIN(ik) GE 0 THEN B[ik]=1
 ;Objects with holes
 IF MAX(B AND (LC GT 0)) EQ 0 THEN C[ik]=ia[k]

```

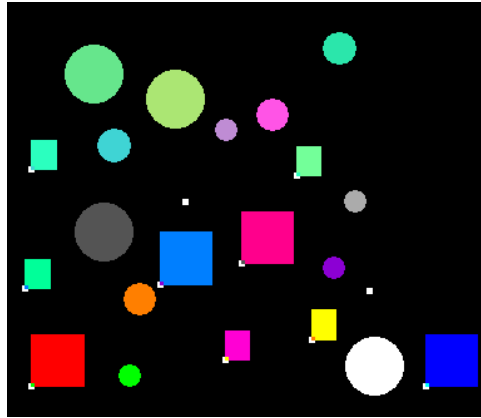
```

END
window,/free,xsize=sa[0],ysize=sa[1],title='Elements without Holes'
TV,C + dilate(Q,replicate(1,5,5))
;Match them up
Print,'Number of circles with holes=',fix(total(Q AND C))
ncircles=nobjects-nsquares
Print,'Number of circles without holes=',ncircles-fix(total(Q AND C))
tv,lct,rr,gg,bb
end

```



Labeled Objects



Objects without holes and markers for square objects are compared to find circular objects without holes.

- Construct a program that will determine the amount of money in the picture `coins1.png` that is shown below. This is a color image. You need to construct ways to tell the difference in the coins. You can use color and size.



Color image `coins1.png` showing a variety of coins of various values.

**Solution** The problem can be solved by using the sizes of the coins to discriminate between them. To measure the coin diameters we can use a monochrome image. This can be converted to a binary image by thresholding. The thresholded image can then be cleaned up by eroding the background noise and closing holes in the coins. It is important that we get solid coin disks. The method used here will not fail if closing the holes in the coins causes them to touch.

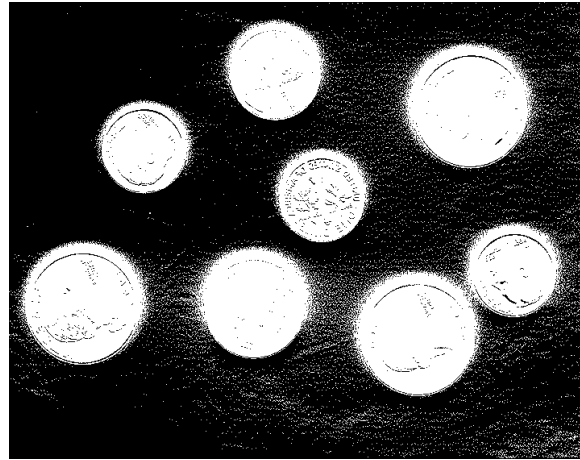
We can then find and remove (most of) each coin by morphological operations. In doing so, we can count each denomination.

```
;Read the coin image and color palette
A=read_image(imgpath+'coins1.png',rr,gg,bb)
;
;Make a RGB version of the image
sa=size(A,/dim)
B=bytarr(3,sa[0],sa[1])
B[0,*,*]=rr[A]
B[1,*,*]=gg[A]
B[2,*,*]=bb[A]
;Make a grayscale image by summing over the three bands
Ag=total(B,1)/3
disp_image,Ag
;
;Threshold the image. Because the background is white,
;use a "less than" threshold.
T=240
At=Ag < T
disp_image,255*At
;Do a closing to clean up the image
S=replicate(1,3,3)
Ac=dilate(erode(At,S),S)
disp_image,255*Ac
;
;The image can be cleaned up by closing holes in the
;coins. A 5x5 structuring element will do.
S1=replicate(1,5,5)
Ac1=erode(dilate(Ac,S1),S1)
disp_image,255*Ac1
```

At this point we have a binary image that we can search for the various coin denominations.



Grayscale



Thresholded (240)

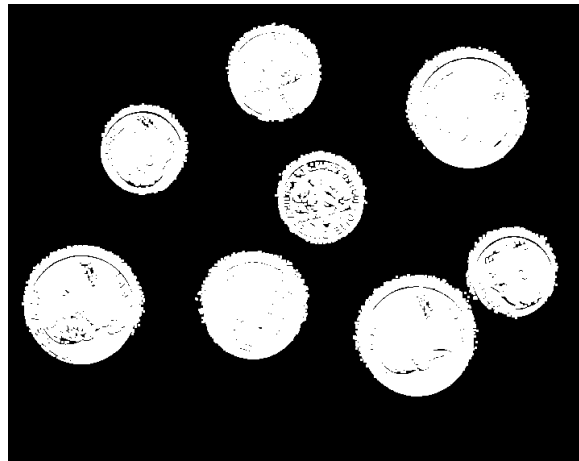


Image of Ac

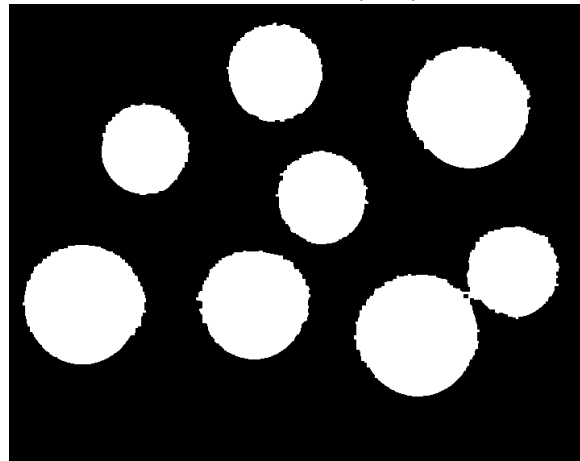


Image of Ac1

An examination of the coin diameters is summarized in the table below. These diameters are used to construct structuring elements.

| Coin    | Diameter |
|---------|----------|
| Quarter | 140      |
| Nickel  | 124      |
| Penny   | 110      |
| Dime    | 102      |

We can construct a structuring element for quarters by using the `disc` function. We will then erode (to locate quarters), dilate (to reconstruct an image of the model for quarters) and remove the quarter image from the set. This will then be repeated for nickels, pennies, and dimes, in that order.

```
Sq=disc(140) LE 70 ;Structuring element for quarters
Aq1=erode(Ac1,Sq)
Aq2=dilate(Aq1,Sq)
Ac2=Ac1 AND NOT Aq2
Aq3=label_region(Aq1,/ALL)
```

```

h=histogram(Aq3)
k=where(h > 0)
nq=n_elements(k)-1 ;Number of quarters
print,format='("Number of quarters found = ",I2)',nq
disp_image,255*Aq1
disp_image,255*Aq2
disp_image,255*Ac2

```

Image Aq1

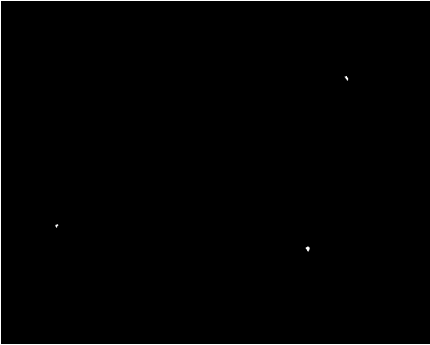


Image Aq2

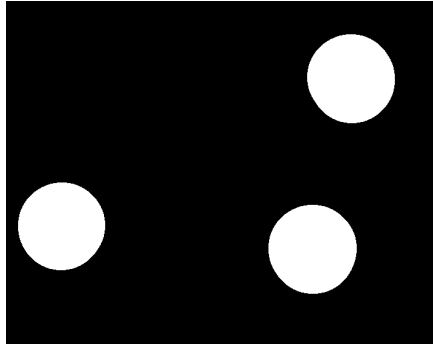
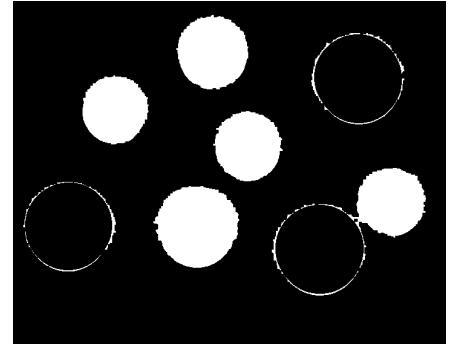


Image Ac2



After erosion of Ac1

After dilation of Aq1

Difference Ac1 and Aq2

Three quarters were found. Now count the nickels

```

Sn=disc(124) LE 62 ;Structuring element for nickels
An1=erode(Ac2,Sn)
An2=dilate(An1,Sn)
Ac3=Ac2 AND NOT An2
An3=label_region(An1,/ALL)
h=histogram(An3)
k=where(h > 0)
nn=n_elements(k)-1 ;Number of nickels
print,format='("Number of nickels found = ",I2)',nn
disp_image,255*An1
disp_image,255*An2
disp_image,255*Ac3

```

Image An1

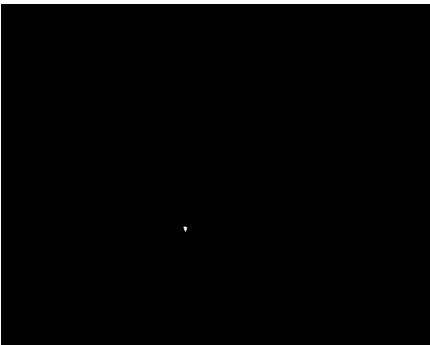


Image An2

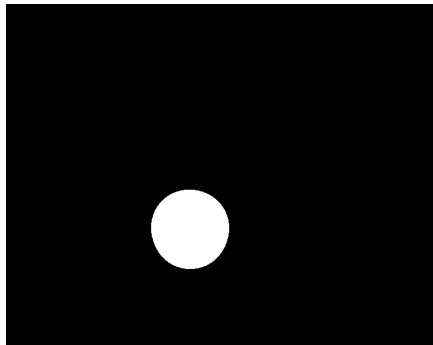
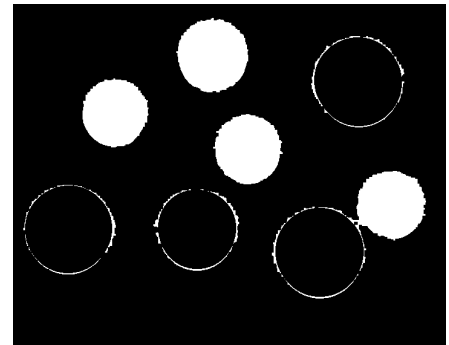


Image Ac3



After erosion of Ac2

After dilation of An1

Difference Ac2 and An2

One nickel was found. Now count the pennies

```
Sp=disc(110) LE 55 ;Structuring element for nickels
Ap1=erode(Ac3,Sp)
Ap2=dilate(Ap1,Sp)
Ac4=Ac3 AND NOT Ap2
Ap3=label_region(Ap1,/ALL)
h=histogram(Ap3)
k=where(h GT 0)
np=n_elements(k)-1 ;Number of pennies
print,format='("Number of pennies found = ",I2)',np
disp_image,255*Ap1
disp_image,255*Ap2
disp_image,255*Ac4
```

Image Ap1

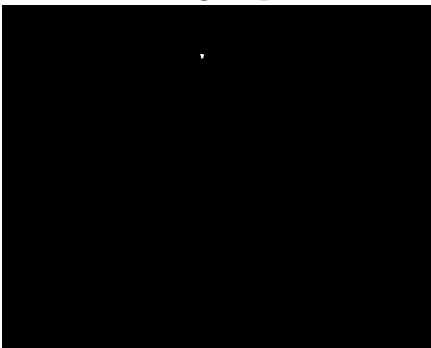


Image Ap2

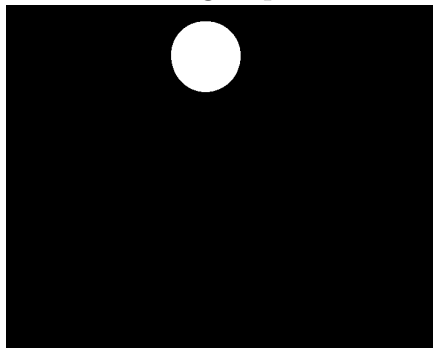
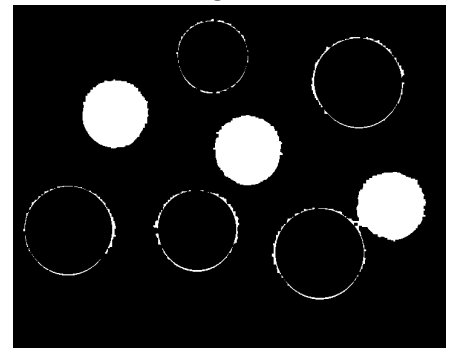


Image Ac4



After erosion of Ac3

After dilation of Ap1

Difference Ac3 and Ap2

One penny was found. Now Count the dimes

```
Sd=disc(102) LE 51 ;Structuring element for nickels
Ad1=erode(Ac4,Sd)
Ad2=dilate(Ad1,Sd)
Ac5=Ac4 AND NOT Ad2
Ad3=label_region(Ad1,/ALL)
h=histogram(Ad3)
k=where(h GT 0)
nd=n_elements(k)-1 ;Number of dimes
print,format='("Number of dimes found = ",I2)',nd
disp_image,255*Ad1
disp_image,255*Ad2
disp_image,255*Ad3
```



Image Ad1

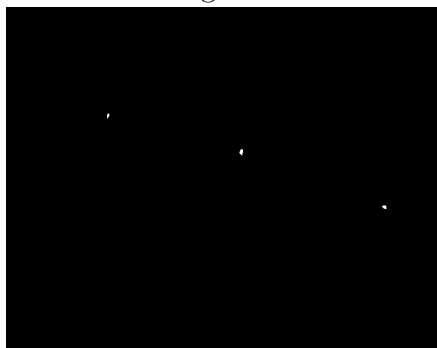


Image Ad2

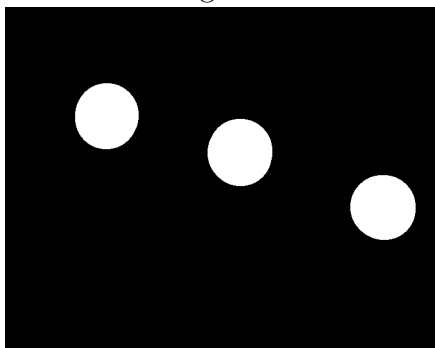
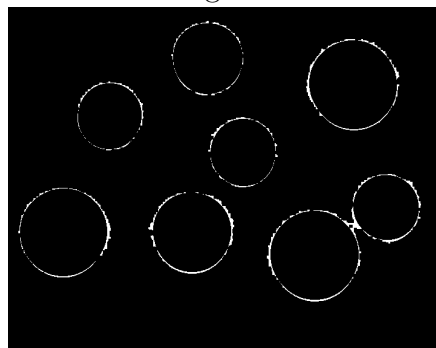


Image Ac5



After erosion of Ac4

After dilation of Ad1

Difference Ac4 and Ad2

Three dimes were found. The total of three quarters, one nickel, three dimes and one penny adds up to \$1.11.