Lecture 10: Hough Circle Transform

Harvey Rhody
Chester F. Carlson Center for Imaging Science
Rochester Institute of Technology
rhody@cis.rit.edu

October 11, 2005

Abstract

Circles are a common geometric structure of interest in computer vision applications. The use of the Hough transform to locate circles will be explained and demonstrated. This is a particular example of the use the Hough transform to search a parameter space.

Circle Hough Transform (CHT)

The Hough transform can be used to determine the parameters of a circle when a number of points that fall on the perimeter are known. A circle with radius R and center (a, b) can be described with the parametric equations

$$x = a + R\cos(\theta)$$
$$y = b + R\sin(\theta)$$

When the angle θ sweeps through the full 360 degree range the points (x, y) trace the perimeter of a circle.

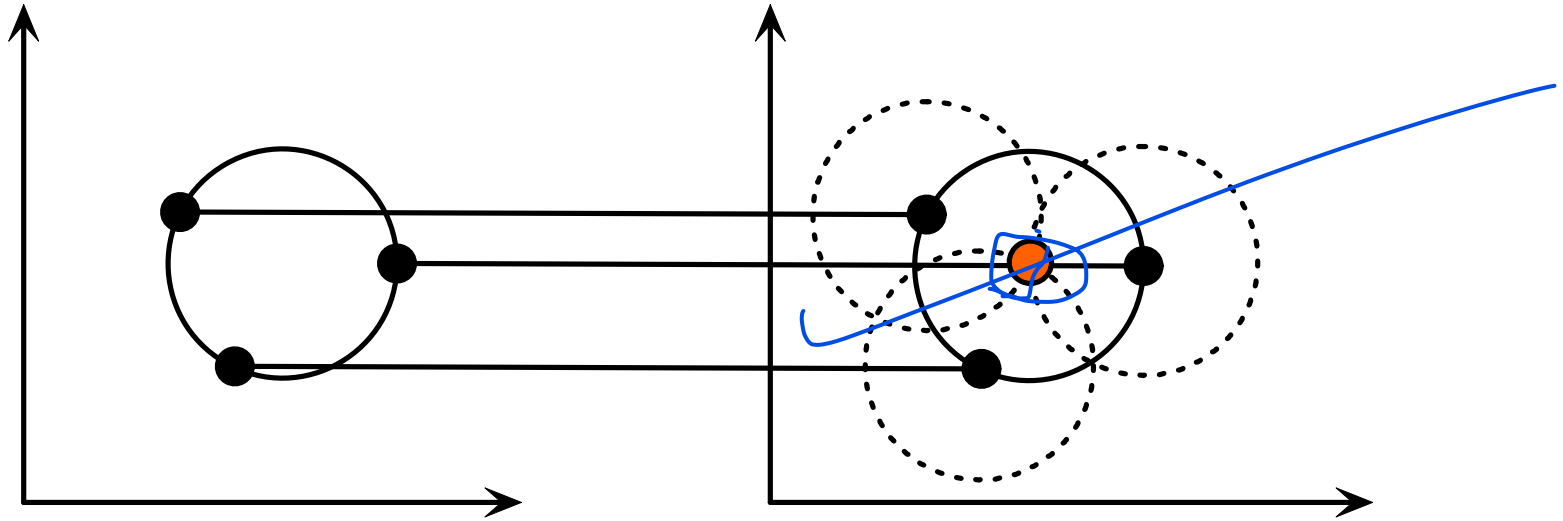
If an image contains many points, some of which fall on perimeters of circles, then the job of the search program is to find parameter triplets (a, b, R) to describe each circle. The fact that the parameter space is 3D makes a direct implementation of the Hough technique more expensive in computer memory and time.

Search with fixed R

If the circles in an image are of known radius R, then the search can be reduced to 2D. The objective is to find the (a, b) coordinates of the centers.

$$\begin{aligned} x &= a + R\cos(\theta) \\ y &= b + R\sin(\theta) \end{aligned}$$

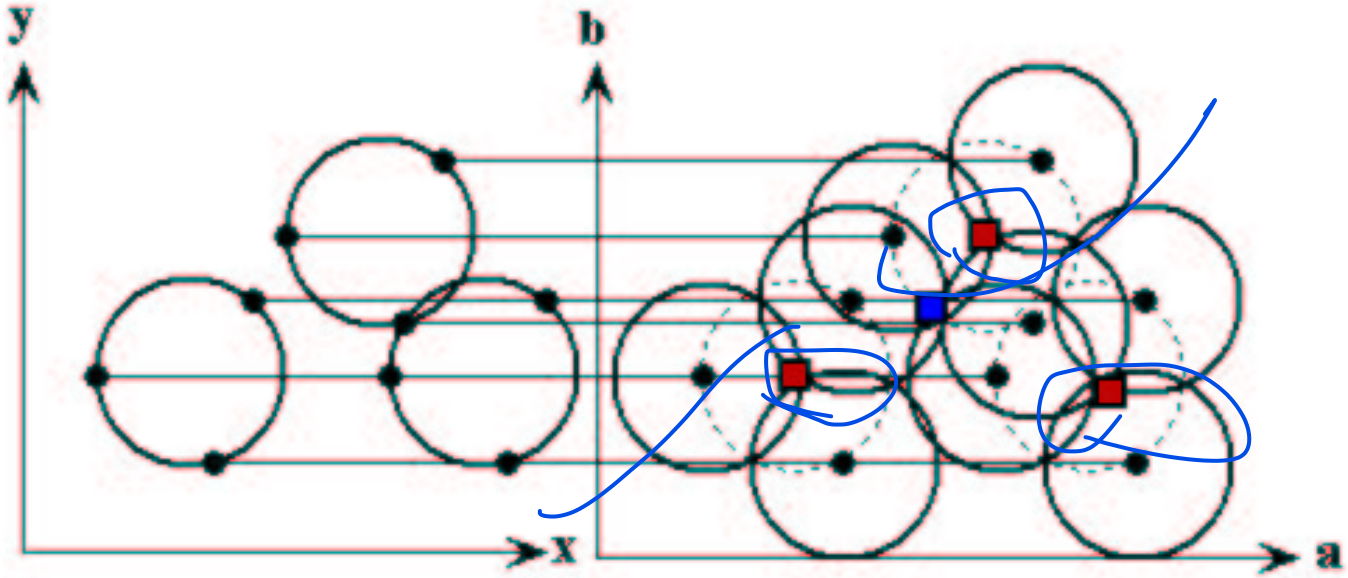
The locus of (a, b) points in the *parameter space* fall on a circle of radius R centered at (x, y). The true center point will be common to all parameter circles, and can be found with a Hough accumulation array.



Each point in geometric space (left) generates a circle in parameter space (right). The circles in parameter space intersect at the (a, b) that is the center in geometric space.

Multiple Circles with known R

Multiple circles with the same radius can be found with the same technique. The centerpoints are represented as red cells in the parameter space drawing. Overlap of circles can cause spurious centers to also be found, such as at the blue cell.
Spurious circles can be removed by matching to circles in the original image.



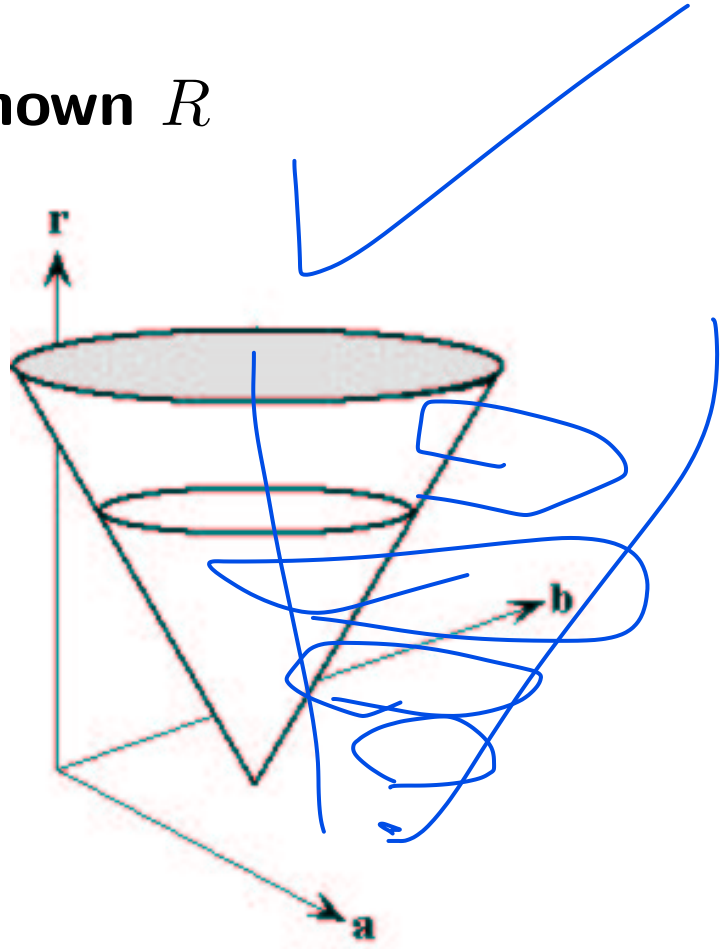
Each point in geometric space (left) generates a circle in parameter space (right). The circles in parameter space intersect at the (a, b) that is the center in geometric space.

Multiple Circles with known R

If the radius is not known, then the locus of points in parameter space will fall on the surface of a cone. Each point (x, y) on the perimeter of a circle will produce a cone surface in parameter space. The triplet (a, b, R) will correspond to the accumulation cell where the largest number of cone surfaces intersect.

The drawing at the right illustrates the generation of a conical surface in parameter space for one (x, y) point. A circle with a different radius will be constructed at each level, r.

The search for circles with unknown radius can be conducted by using a three dimensional accumulation matrix.

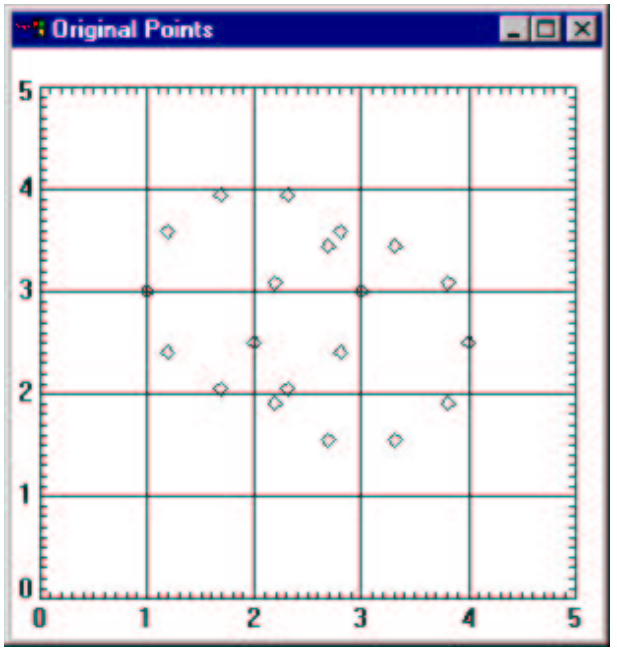


Example 1

Points on overlapping circles of known radius.

A set of 20 points on circles of radius 1 and centers [2,3] and [3,2.5] were generated. These are displayed at the right.

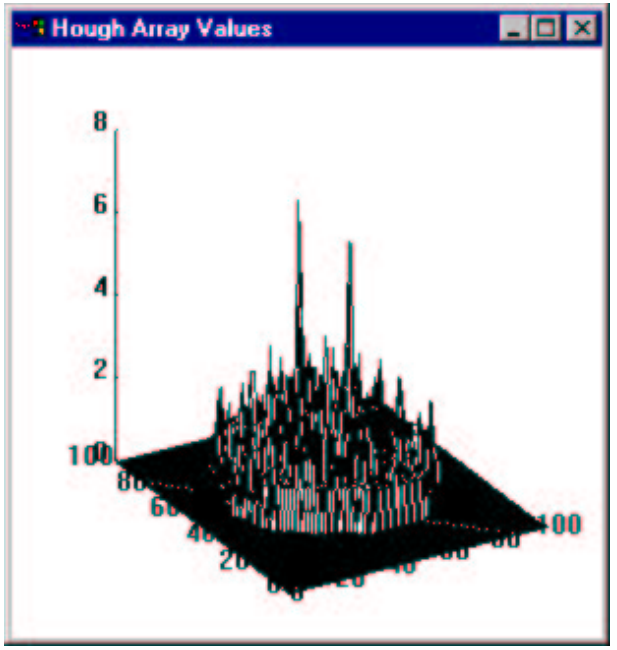
This data set was submitted to the CHT program with the radius R=1 given.



Example 1 (cont)

The CHT accumulation matrix is shown in a surface plot at the right. Two peaks are very clear. These correspond to the locations of the centers of the circles.

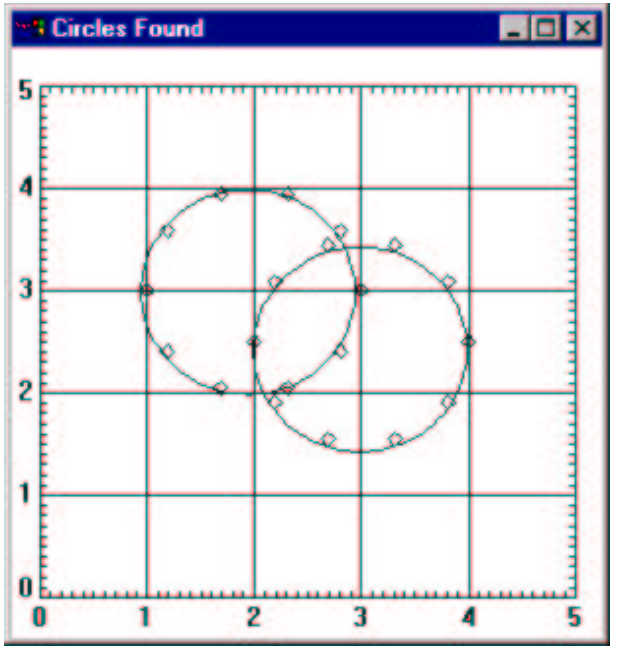
The circles were selected using a threshold value of thresh=7.



Example 1 (cont)

The data for the circles is shown below. The circles are plotted from this data over the given data points in the figure below right.

N	R	C_x	C_y
1	1.00	1.94	2.99
2	1.00	2.99	2.43



Example 2

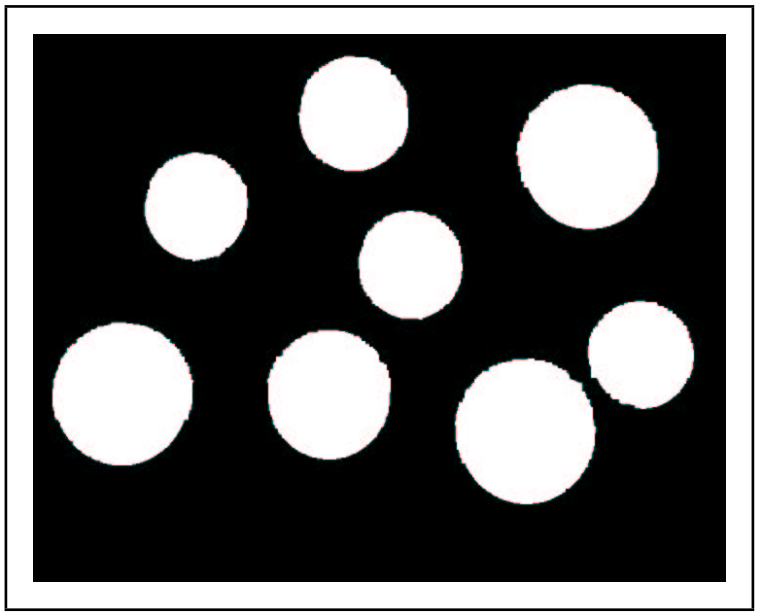
Search for circles in the coins1 image.

Since the coins are round, we would expect to be able to find circles to match their edges. This example will illustrate a sequence of pre-processing steps and then a CHT search.

Example 2 (cont)

We need to find the edges in the original image. The first step is to convert it to a binary image with a threshold operation. That is followed by a morphological opening to remove background noise. The opening is followed by a closing to remove small holes.

The result is shown at the right.



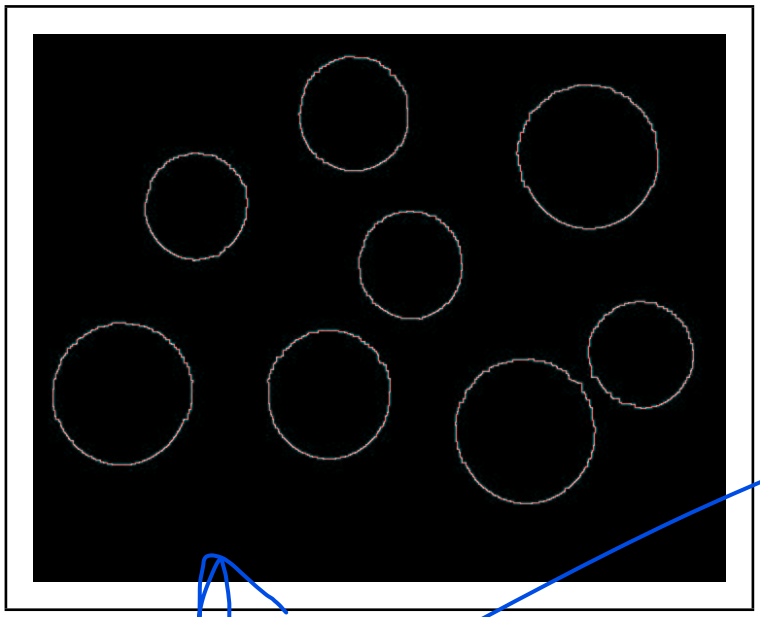
```
A=READ_IMAGE('coins1.png',rr,gg,bb)
G=BytScl(Float(rr[A])+ $
    Float(gg[A])+Float(bb[A]))
thresh=180
BW=G LE 235
S=Replicate(1,3,3)
IM1=DILATE(ERODE(BW,S),S)
S=Replicate(1,13,13)
IM2=ERODE(DILATE(IM1,S,S)
```

Example 2 (cont)

The edges are located by a morphological boundary operation

```
S=Replicate(1,3,3)
IM3=IM2 AND NOT ERODE(IM2)
```

The result is shown at the right.

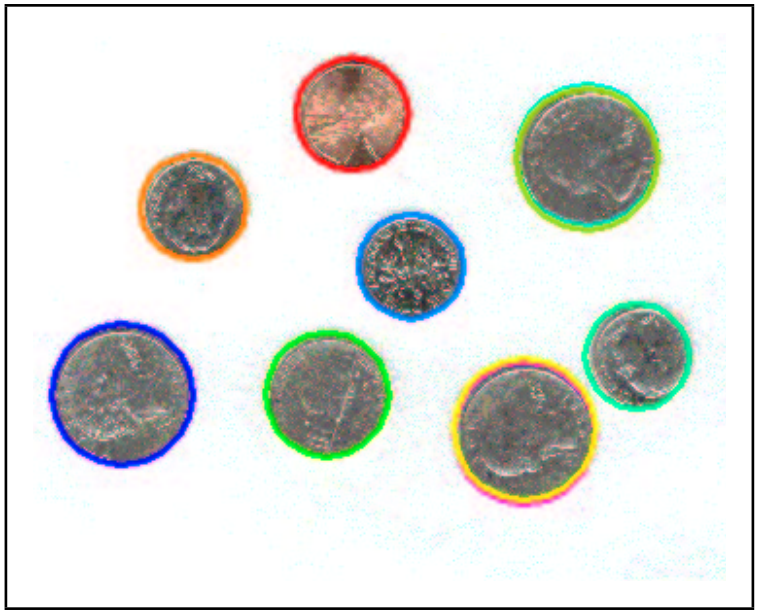


Example 2 (cont)

The search program was given set up to search for circles with specified radii. The radius parameters matched the known coin values.

A search parameter was set to find 10 circles. This allows some latitude in the process. Results are shown below and in the image at the right.

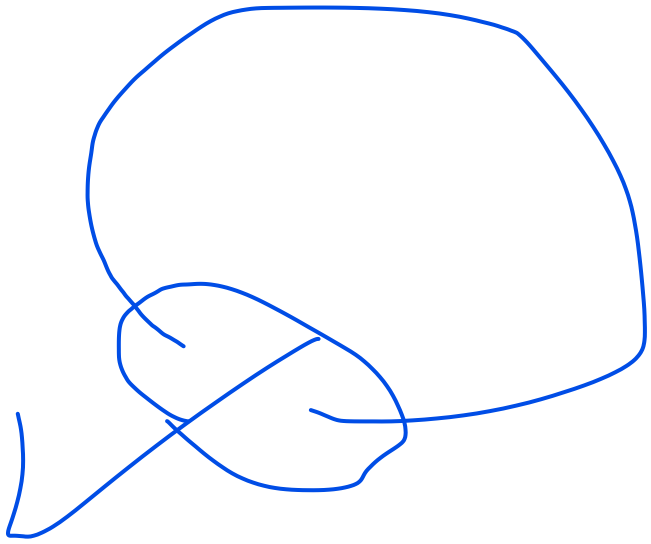
k	R	C_x	C_y	Count
1	58	327	479	394
2	64	301	190	392
3	73	90	190	379
4	73	569	436	369
5	73	504	151	331
6	73	504	155	310
7	54	164	383	303
8	73	569	433	287
9	54	620	230	269
10	54	388	322	268



Example 3–Partial Circles

Finding circles that are partially hidden is possible with the CHT if enough of the boundary is visible.

Shown at the right is an image in which some of the objects are partially occluded by others.



Example 3 (cont)

We need to find the edges in the original image. The first step is to convert it to a binary image with a threshold operation. That is followed by a morphological opening to remove background noise. The opening is followed by a closing to remove small holes.

The result is shown at the right.

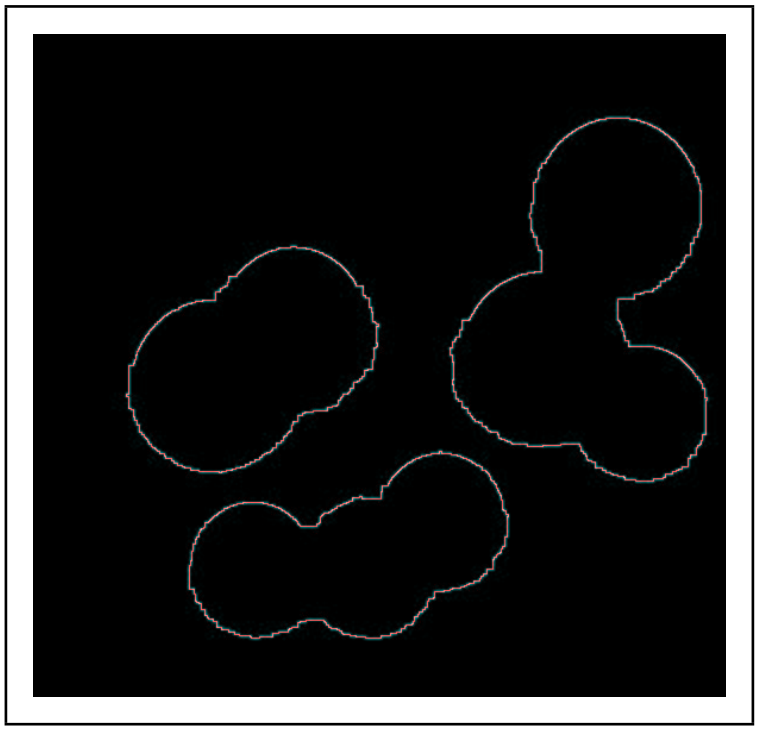
```
A=READ_IMAGE('coins2.png',rr,gg,bb)
G=BytScl(Float(rr[A])+ $
    Float(gg[A])+Float(bb[A]))
thresh=180
BW=G LE 235
S=Replicate(1,3,3)
IM1=DILATE(ERODE(BW,S),S)
S=Replicate(1,13,13)
IM2=ERODE(DILATE(IM1,S,S)
```

Example 3 (cont)

The edges are located by a morphological boundary operation

```
S=Replicate(1,3,3)
IM3=IM2 AND NOT ERODE(IM2)
```

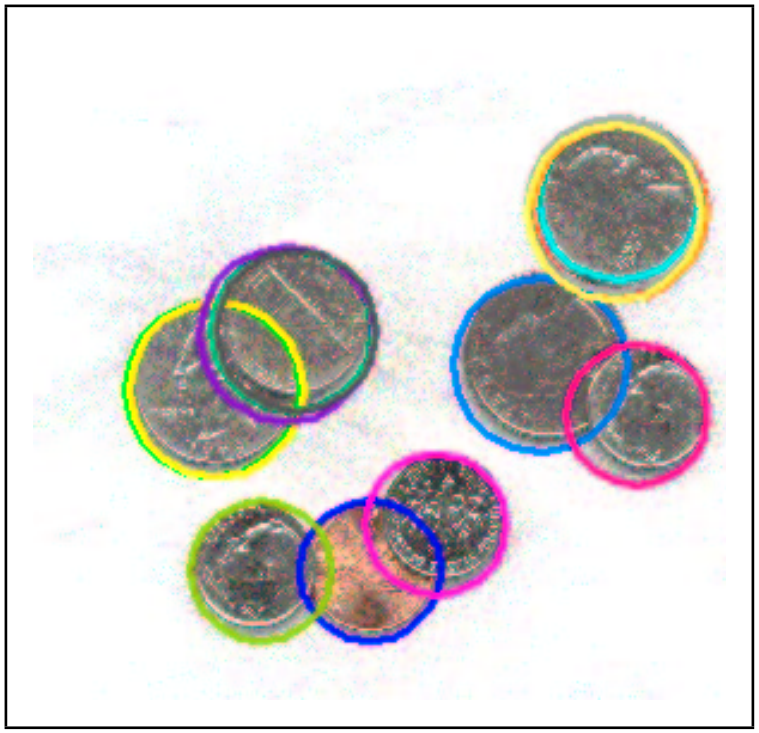
The result is shown at the right.



Example 3 (cont)

The search program was given set up to search for 15 circles with specified radii. Results are shown below and in the image at the right.

k	R	C_x	C_y	Count
1	73	489	409	266
2	73	149	258	212
3	59	281	106	196
4	67	489	416	192
5	59	335	145	187
6	73	153	258	184
7	73	489	406	183
8	59	189	106	174
9	67	215	307	171
10	73	423	279	169
11	73	211	304	167
12	59	503	237	165
13	67	219	307	163
14	73	485	413	159
15	73	485	406	158



Demonstration: coinDemo1

```
;Get the image and display it. Set and restore
;the color table.

tvlct,r0,g0,b0,/get
A=read_image(imgpath+'coins1.png',rr,gg,bb)
sa=size(A,/dim)

tvlct,rr,gg,bb
disp_image,A,title='Original',xp=0,yp=0
win1=!D.WINDOW ;remember the window index
tvlct,r0,g0,b0

G=(float(rr[A])+float(gg[A])+float(bb[A]))/3
disp_image,G,title='Grayscale',xp=640,yp=0

;Make a binary image of the objects on a
;black background
thresh=235
BW=G LE thresh
S1=replicate(1,3,3)
IM1=morph_open(BW,S1)
```

```
S2=replicate(1,13,13)
IM2=morph_close(IM1,S2)

disp_image,bytscl(IM1),xp=0,yp=200,title='Opened'
disp_image,bytscl(IM2),xp=640,yp=200,title='Closed'

IM3=IM2 AND NOT erode(IM2,S1)
disp_image,bytscl(IM3),xp=0,yp=400,title='Outline'
win2=!D.WINDOW

R=[53,58,64,73]
k=where(IM3 GT 0)
xpts=k mod sa[0]
ypts=k/sa[0]
P=circlehoughlink(xpts,ypts,RADIUS=R,NCIRCLES=12,$
    THRESHOLD=300,XBINS=200,YBINS=200)
wset,win2
tek_color
phi=findgen(101)/50*!pi
for k=0,n_elements(P.R)-1 do begin &$
    rho=P[k].R &$
    x=P[k].Cx+rho*cos(phi) & y=P[k].Cy+rho*sin(phi) &$
```

```
    plot,x,y,/noerase,color=k+2,xmargin=[0,0],ymargin=[0,0], $
        xtickv=[0,sa[0]],ytickv=[0,sa[1]],xticks=1,yticks=1 &$
endfor

TVLCT,r0,g0,b0
```

Demonstration: CoinDemo2

```
;Get the image and display it. Set and restore
;the color table.

tvlct,r0,g0,b0,/get
A=read_image(imgpath+'coins2.png',rr,gg,bb)
sa=size(A,/dim)

tvlct,rr,gg,bb
disp_image,A,title='Original',xp=0,yp=0
win1=!D.WINDOW ;remember the window index
tvlct,r0,g0,b0

G=(float(rr[A])+float(gg[A])+float(bb[A]))/3
disp_image,G,title='Grayscale',xp=640,yp=0

;Make a binary image of the objects on a
;black background
thresh=235
BW=G LE thresh
S1=replicate(1,3,3)
IM1=morph_open(BW,S1)
```

```
S2=replicate(1,13,13)
IM2=morph_close(IM1,S2)

disp_image,bytscl(IM1),xp=0,yp=200,title='Opened'
disp_image,bytscl(IM2),xp=640,yp=200,title='Closed'

IM3=IM2 AND NOT erode(IM2,S1)
disp_image,bytscl(IM3),xp=0,yp=400,title='Outline'
win2=!D.WINDOW

R=[53,58,64,73]
k=where(IM3 GT 0)
xpts=k mod sa[0]
ypts=k/sa[0]
P=circlehoughlink(xpts,ypts,RADIUS=R,NCIRCLES=8,$
    THRESHOLD=200,XBINS=200,YBINS=200)
wset,win2
tek_color
phi=findgen(101)/50*!pi
for k=0,n_elements(P.R)-1 do begin &$
    rho=P[k].R &$
    x=P[k].Cx+rho*cos(phi) & y=P[k].Cy+rho*sin(phi) &$
```

```
    plot,x,y,/noerase,color=k+2,xmargin=[0,0],ymargin=[0,0], $
        xtickv=[0,sa[0]],ytickv=[0,sa[1]],xticks=1,yticks=1 &$
endfor

TVLCT,r0,g0,b0
```