Example 1 - lines

```
clear all
close all

I = imread('hallway.jpg');
imshow(I,[]);

E = edge(I, 'canny');
figure, imshow(E);

[H, theta, rho] = hough(E);
peaks = houghpeaks(H, 10);

% Draw each line
figure, imshow(I,[]);
for i=1:size(peaks,1)
    % Extract rho, theta for this line
    r = rho(peaks(i,1));
    t = theta(peaks(i,2));

    % Equation of the line is r = x cos(t) + y sin(t), or
    %    y = (r - x*cos(t))/sin(t)
    x0 = 1;
    y0 = (r - x0*cosd(t))/sind(t);
    x1 = size(I,2);
    y1 = (r - x1*cosd(t))/sind(t);

    line([x0 x1], [y0 y1], 'Color', 'r');
end
```

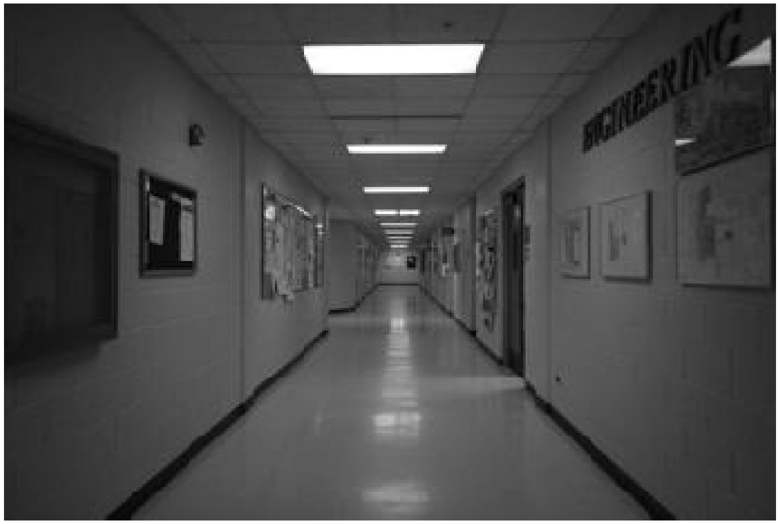


image “hallway.jpg”

Example 1 (cont)

```
clear all
close all

I = imread('hallway.jpg');
imshow(I,[]);

E = edge(I, 'canny');
figure, imshow(E);

[H, theta, rho] = hough(E);
peaks = houghpeaks(H, 10);

% Use Matlab's Hough lines
lines = houghlines(E, theta, rho, peaks);

figure, imshow(I);
for k = 1:length(lines)
   xy = [lines(k).point1; lines(k).point2];
   line(xy(:,1),xy(:,2),'LineWidth',1.5,'Color','g');
end
```

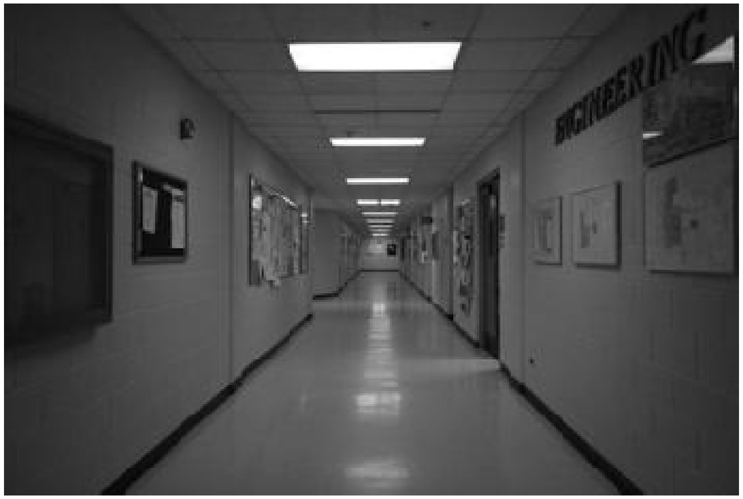


image “hallway.jpg”

Example 2 - circles

```
% Find circles using Hough transform.
clear all
close all

I = imread('coins.png');
imshow(I,[]);

% Add noise and occlude some of the coins
I(100:120, :) = 100;
I = imnoise(I, 'gaussian', 0, 0.001);
figure, imshow(I,[]);

[centers, radii] = imfindcircles(I, ...
    [20, 35], ...       % Range of radii to search for
    'ObjectPolarity', 'bright', ...   % Circles are brighter than bkgnd
    'Sensitivity', 0.9);       % Higher values yield more circles

figure, imshow(I, []);
for j=1:size(centers,1)
    r = radii(j);
    x = centers(j,1);
    y = centers(j,2);
    rectangle('Position', [x-r y-r 2*r 2*r], 'EdgeColor', 'r', ...
        'Curvature', [1 1], 'LineWidth', 2);
end
```

Example 3 – circles in video

```
clear all
close all

movie = VideoReader('sphero1.wmv');
images = read(movie);    % get all images
nImg = size(images,4);      % Number of images read
fprintf('Read in %d images from video file\n', nImg);

% Suppress warnings about low accuracy because of small radii.
warning('off', 'all');

for i=1:2:nImg
    I = images(:,:,:,i);     % Get next image
    G = rgb2gray(I);

    [centers, radii] = imfindcircles(G, [5, 20]);

    imshow(I, []), title(sprintf('%d', i));
    for j=1:size(centers,1)
        r = radii(j);
        x = centers(j,1);
        y = centers(j,2);
        rectangle('Position', [x-r y-r 2*r 2*r], 'EdgeColor', 'r', ...
            'Curvature', [1 1], 'LineWidth', 2);
    end
    drawnow;
end
```

Matlab code and images from Grant Latham

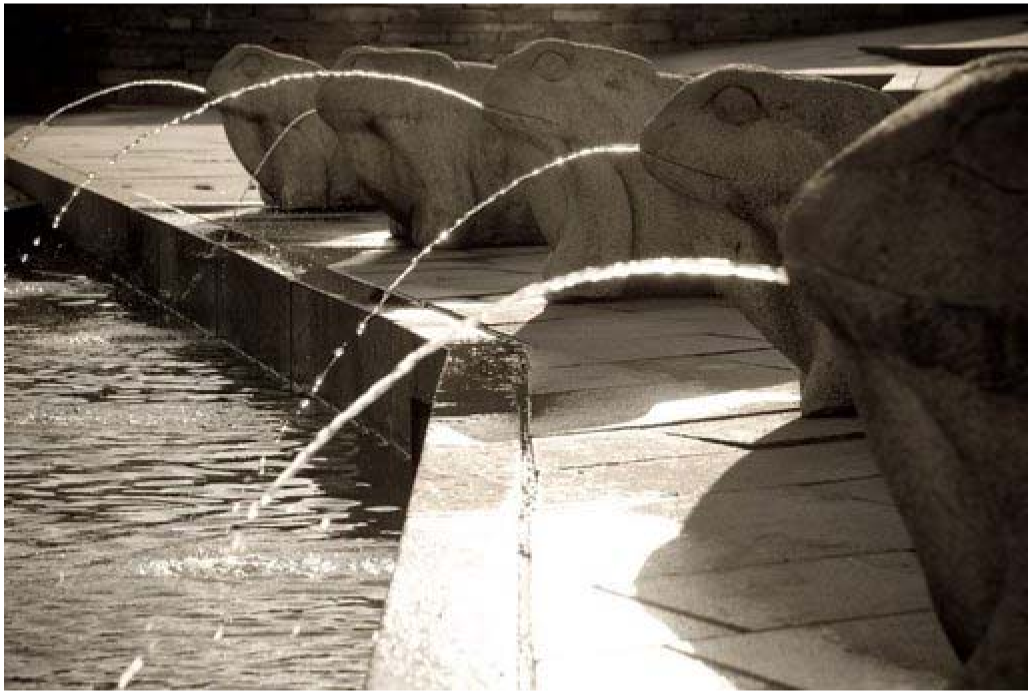
Example 4 –parabolas

- A parabola centered at (x0,y0) has the equation

$$y - y_0 = a\left(x - x_0\right)^2$$

- So we need to search for three parameters: x0,y0,a

- Can limit range of “a” to search over



Grenouilles.jpg

Pseudocode

```
Input a binary edge image E(x,y)
Initialize accumulator array A(i,j,k) to zeros

for all values of (x,y)
  if E(x,y) == 1
    for all values of a between amin and amax
      for all values of x0
        Compute $y_0 = y - a\left(x - x_0\right)^2$

        Increment A(i,j,k) where (i,j,k) corresponds to the cell
          associated with $\left(x0_i, y0_j, a_k\right)$
        end
    end
  end
End

Search for peaks in A(i,j,k) – the corresponding values of
are the parameters of the detected parabolas
```

```
% Find parabolas in binary image I(N,M).
% A parabola is y=rx^2
clear all
close all

I = rgb2gray(imread('Grenouilles.jpg'));
imshow(I,[]);
[N,M] = size(I);

[E,thresh] = edge(I, 'canny', 0.45);
figure, imshow(E,[]);

rvals = [ 0.003, 0.004, 0.005, 0.006, 0.007, 0.008 ];
R = length(rvals);   % number of sizes to try

% Fill accumulator array A(N,M,R)
A = zeros(N,M,R);
[yIndex xIndex] = find(E);      % get edge points
for cnt = 1:length(xIndex)
    for r=1:R
        for x0 = 1:M
            y0 = yIndex(cnt)-rvals(r)*(xIndex(cnt)-x0)^2;
            y0 = round(y0);
            if y0 < N & y0 >= 1
                A(y0,x0,r) = A(y0,x0,r)+1;
            end
        end
    end
end

figure, imshow(A(:,:,round(R/2)),[]);
title(sprintf('A slice of the accumulator array, for r=%f', ...
    rvals(round(R/2)) ));
pause
```

```
% Find the local maxima in a 3D neighborhood
Amax = imdilate(A, ones(20,20,4));

% We want those places where A = AMax and A > thresh
thresh = 95;
Apeaks = (A==Amax) & (A > thresh);

% Get indices of the peaks (the nonzero points).
% These are the indices of the array, which is treated
% like a one dimensional array.
indices1D = find(Apeaks);

% The array is actually three dimensional, not one dimensional.
% This function will calculate the equivalent indices of the three
% dimensional array.
[rowIndices,colIndices,depthIndices] = ...
    ind2sub(size(Apeaks), indices1D);


figure, imshow(E,[]);
for i=1:length(rowIndices)
    x0 = colIndices(i);
    y0 = rowIndices(i);
    r0 = rvals( depthIndices(i) );
    for x=1:M
        y = round(y0 + r0*(x-x0)^2);
        if y<=N && y >= 1
            rectangle('Position', [x y 1 1], ...
            'EdgeColor', 'r');
        end
    end
end
```