```
display(['Image Processing'])
clear;
load im30; %image taken from the whiffle ball video
figure(2); %name of image
clf; %clears any figures before showing image
imshow(im30); %Upload and showing the image
hold on; %Permentantly holds the image
axis on; %Permenantly holds the axis values
[c r] = imfindcircles(im30,[13 100],'sensitivity',.91,'EdgeThreshold',.1); % c holds the pixel coordinates of the ball's center. r is the approximation of the radiu
viscircles(c,r); %adds circles with specified centers and radii to the current axes. CENTERS is a 2-column matrix with
    % X-coordinates of the circle centers in the first column and Y-coordinates in the second column. RADII is a vector which specifies
    % radius for each circle. By default the circles are red.
[x y]=ginput(4); % gets N points from the current axes and returns the X- and Y-coordinates in length N vectors X and Y.  The cursor can be positioned using a mouse
distance=zeros(4); %creates an NxN array of zeros
for i = 1:4
    for j = 1:4
        distance(i,j) = sqrt((x(i)-x(j))^2+(y(i)-y(j))^2); % calculates the distance between centers of the balls.
    end
end
disp(distance); %Displays distance to centers



display(['Video Tracking'])
v=VideoReader('wiffleBalls.mov'); %opens the video wiffleBalls
[c r] = imfindcircles(im30,[13 100],'sensitivity',.91,'EdgeThreshold',.1);
prevCenter = c; %the initial position of the center of the ball
MovingBall = zeros(47,1); %47 frames in the video
k = 1;
while hasFrame(v)==true %goes frame by frame to track the positioning of the ball and the velocity
    im = readFrame(v); % reads the next aviable video frame from the file
    figure(3); %name this image, image 3. Constantly gets updated until there are no more frames left
    imshow(im);
    hold on;
    axis on;
    [c r] = imfindcircles(im,[25 70],'sensitivity',.92,'EdgeThreshold',.065); %adjust imfindcircles so the circle finder moves with the ball
    currentcenter = c; %preparing to compare center positions
    viscircles(c,r);
    for i = 1:size(prevCenter,1)
        for j = 1:size(c,1)
            dist(i,j) = sqrt((currentcenter(j,1)-prevCenter(i,1))^2+(currentcenter(j,2)-prevCenter(i,2))^2);
        end %calculates the distance between the initial frame and the one after it
    end
    prevCenter = currentcenter; %shift the data points
    centerMoved = zeros(size(c,1),1); %creates a 4x1 zero array (4 balls)
    for i= 1:size(dist,1) %1-4
        centerMoved(i) = min(dist(i,:));
    end
    MovingBall(k) = max(centerMoved); %the moving ball has the largest mini. change
    velocity = MovingBall*v.framerate*.005; %conversion factor to get velocity from frame rate
    disp(velocity(k));
    disp('ft/sec');
    k = k+1;
end
figure(4);
MovingBall(1,1) = 0;
velocity (1,1) = 0;
position = k/v.framerate;
t= linspace(1,position,k);
plot (velocity);


%Notes
%[c r] = imfindcircles(im30,[10 100])
    %storing x coords. in c and y coords. in r (done in pixels).
    %[10 100] is the min and max radius. The estimated radii, in pixels, for he circles are
    %returned in the column vector RADII (r=radii, c=center

% 'Sensitivity '  - Specifies the sensitivity factor in the range [0 1]
%                       for finding circles. A high sensitivity value leads
%                       to detecting more circles, including weak or
%                       partially obscured ones, at the risk of a higher
%                       false detection rate. Default value: 0.85
%
%  'EdgeThreshold' - A scalar K in the range [0 1], specifying the gradient
%                    threshold for determining edge pixels. K = 0 sets the
%                    threshold at zero-gradient magnitude, and K = 1 sets
%                    the threshold at the maximum gradient magnitude in
%                    the image. A high EdgeThreshold value leads to
%                    detecting only those circles that have relatively
%                    strong edges. A low EdgeThreshold value will, in
%                    addition, lead to detecting circles with relatively
%                    faint edges. By default, imfindcircles chooses the
%                    value automatically using the function GRAYTHRESH.

%ginput Graphical input from mouse.
%     [X,Y] = ginput(N) gets N points from the current axes and returns
%     the X- and Y-coordinates in length N vectors X and Y.  The cursor
%     can be positioned using a mouse.  Data points are entered by pressing
```

```
%     a mouse button or any key on the keyboard except carriage return,
%     which terminates the input before N points are entered.
```

```
Image Processing
         0  127.7693  184.9459  331.2658
  127.7693         0  234.7637  366.7642
  184.9459  234.7637         0  146.3216
  331.2658  366.7642  146.3216         0

Video Tracking
   17.3383

ft/sec
    0.0667

ft/sec
    0.1656

ft/sec
    0.1099

ft/sec
    0.0686

ft/sec
    0.0898

ft/sec
    0.1322

ft/sec
    0.1704

ft/sec
    0.0938

ft/sec
    0.1005

ft/sec
    0.1785

ft/sec
    0.1212

ft/sec
    0.1604

ft/sec
    0.1377

ft/sec
    0.1222

ft/sec
    0.1089

ft/sec
    2.6916

ft/sec
    3.9583

ft/sec
    3.2030

ft/sec
    2.8889

ft/sec
    2.3347

ft/sec
    2.4803

ft/sec
    2.4642

ft/sec
    2.1981

ft/sec
    1.9026

ft/sec
    2.4131

ft/sec
    1.8057
```

ft/sec
    2.1446

ft/sec
    2.3911

ft/sec
    2.0701

ft/sec
    1.9925

ft/sec
    2.2758

ft/sec
    1.8914

ft/sec
    2.2555

ft/sec
    2.3963

ft/sec
    1.6087

ft/sec
    2.3548

ft/sec
    2.1051

ft/sec
    2.2483

ft/sec
    2.1195

ft/sec
    1.9943

ft/sec
    2.0940

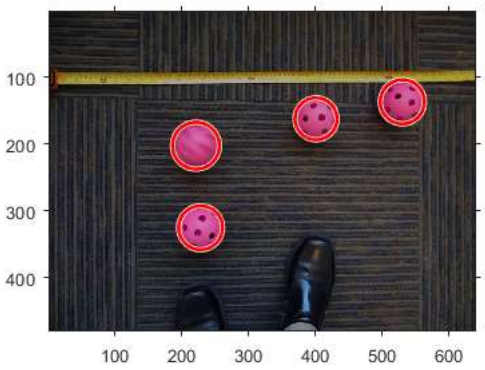
ft/sec
    1.9997

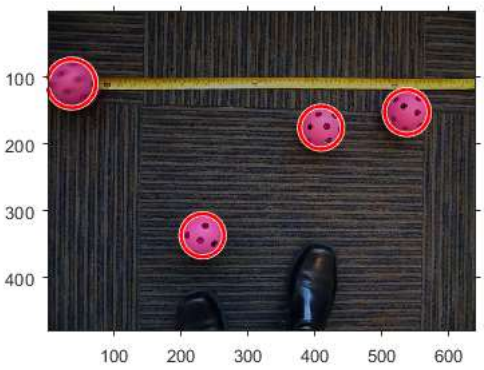
ft/sec
    1.8094

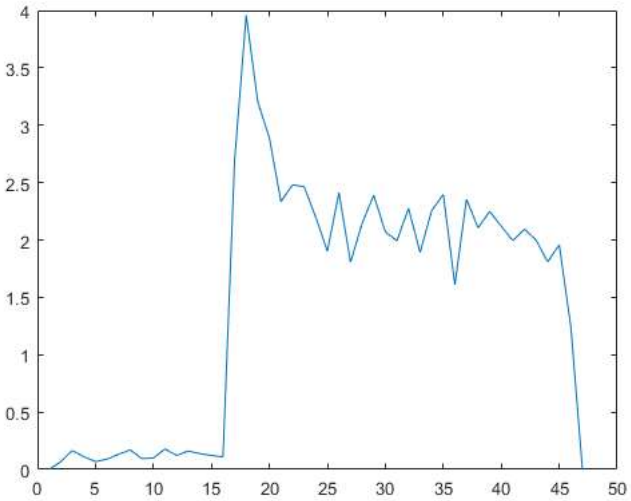
ft/sec
    1.9572

ft/sec
    1.2448

ft/sec







Published with MATLAB® R2021a