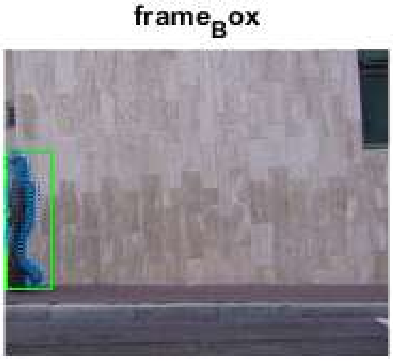
```
close all
clear;
clc;
% video resource and constant parameters
videoSource = VideoReader('F:\dataset\datasets\walk\daria_walk.avi');%F:\dataset\datasets\
walk
width = videoSource.Width;
height =videoSource.Height;
Threshold = 0.02;
CloseSize = 8;
desiredAngles = 5:5:360;
heiwid = numel(videoSource);
% allDistance = zeros(heiwid,length(desiredAngles));% creat a length(i) array with zeros
boundary = ones(height,width);

%
% % import data
% addpath('libsvm-3.25\matlab')
% % import libsvm and data set
% addpath('F:\github_lib\Opticalflow4HAR\libsvm-3.25\matlab');

%
% train = load('train_ratio.csv') ;
% test = load ('test_ratio.csv');
% % generate random number for divide data into training and test set.
% %n = randperm(size(data,1));
% %t = randperm(size(test,1));
% %training set
% train_matrix = train(:,1:9); % training data 1-70,6 columns
% train_label = train(:,10); % training label at 7 column.
% % data normalization
% [train_matrix, PS] = mapminmax(train_matrix');
% train_matrix = train_matrix';
% % test_matrix = mapminmax('apply',test_matrix', PS);
% % test_matrix = test_matrix';
%
% model = svmtrain(train_label,train_matrix,'-s 0 -t 2 -c 1.2 -g 2.8 -b 1'); % -v-s 0 -t 2
 -c 1.2 -g 2.8'




% Guassian mixture model :foreground
%creat a detector object
detector = vision.ForegroundDetector( ...
    'NumTrainingFrames',5, 'InitialVariance', 30*30);
% Perform blob analysis.
blob = vision.BlobAnalysis(...
       'CentroidOutputPort', false, 'AreaOutputPort', false, ...
       'BoundingBoxOutputPort', true, ...
       'MinimumBlobAreaSource', 'Property', 'MinimumBlobArea', 800,...
  'MaximumCount',3);
% insert a bounder
shapeInserter = vision.ShapeInserter('BorderColor','Custom');


% optical flow parameters
opticFlow = opticalFlowHS('Smoothness',1, 'MaxIteration', 10, 'VelocityDifference', 0);
opticalBG = ones(height, width)*255;
```

```
% frameLogical
frameLogical = ones(height,width);

%access frame
while hasFrame(videoSource)
    frameRGB = readFrame(videoSource);
    frameGray = rgb2gray(frameRGB);

    flow = estimateFlow(opticFlow, frameGray); % estimateFlow based on shape
    % fmask = detector(frameGray);% mask
         % calculate frameLogical binary frame for boundaries point.
    for i = 1:height
        for j = 1:width
            if(sqrt(flow.Vx(i, j)^2 + flow.Vy(i, j)^2) <= Threshold)%threshold=0.02
                frameLogical(i, j) = 0;
            else
                frameLogical(i, j) = 255;% flow.Vx()...>threshold
            end
        end
    end
    se = strel('square', CloseSize);% morphlogical
    % frameLogical = logical(frameLogical);% binarize frame logical type

    frameLogical = imclose(frameLogical, se);
    frameLogical =logical(frameLogical);
    bbox = step(blob, frameLogical); % boundary
    out = insertShape(frameRGB,'Rectangle',bbox,'color','green');% output with boundary on
 frame

    % calculate center of gravity of the foreground
    [labelImage, numberOfImage] = bwlabel(frameLogical, 8);
    blobMeasurements = regionprops(labelImage, 'Centroid');
    yCenter = blobMeasurements(1).Centroid(1);
    xCenter = blobMeasurements(1).Centroid(2);
    if xCenter>height && yCenter>width
        disp('out of iamge...');
        continue;
    end

    % boundaries point from framlogical.
    [B,L] = bwboundaries(frameLogical,'noholes');
    boundaries=B;% shape boundary coordinate

    boundaries = boundaries{1};
    Boun_num = size(boundaries);
    % boundaries of shapes all
    xb = boundaries(:,1);
    yb = boundaries(:,2);



    % calculate angle of every boundaries point and distance
    angles = atan2d((yb-yCenter),(xb-xCenter))+180;% every angles of boundaries point

    position_coordinate = [xb,yb,angles];
    %distances = sqrt((xb-xCenter).^2+(yb-yCenter).^2);



    % maybe more than 1 index point with the same angle
    [uniqueAngles, ia, ic]= unique(angles);% ia index of original vector,
    uniquexb= xb(ia);
    uniqueyb = yb(ia);
    %uniqueDistances = distances(ia);
    uniqueAngles = [uniqueAngles(end)-360; uniqueAngles; uniqueAngles(1) + 360];
    uniquexb = [uniquexb(end); uniquexb; uniquexb(1)];
    uniqueyb = [uniqueyb(end); uniqueyb; uniqueyb(1)];
    desiredxb = interp1(uniqueAngles, uniquexb,desiredAngles);
    desiredyb = interp1(uniqueAngles, uniqueyb,desiredAngles);
    %uniqueDistances = [uniqueDistances(end); uniqueDistances; uniqueDistances(1)];
    % desiredDistances = interp1(uniqueAngles, uniqueDistances,desiredAngles);
    %allcoordinate(:,1) = desiredDistances;
    allcoordinate(:,1) = desiredxb;
    allcoordinate(:,2) = desiredyb;

xc=round(xCenter);
yc=round(yCenter);%(x,y)
xe=round(desiredxb);
ye=round(desiredyb);% desiredyb
count = 1;
result =[];
for inx = 1:72
    if all(xe(inx)==xc)&& all(yc==ye(inx))
        disp('the same point with centroid');
    continue;
    end
    if all(xc==(xe(inx))) || all(yc==(ye(inx)))
        intersection_space = zeros(1,9);
        intersection_space(1,1)= xe(inx);
        intersection_space(1,2)= ye(inx);
        intersection_space(1,3)= sqrt((xe(inx)-xCenter).^2+(ye(inx)-yCenter).^2);
        result= [result,intersection_space];
        continue;
    end
    if (xe(inx)>xc)
        steps = 1;
    else
        steps = -1;
    end
     k =((ye(inx)-yc)/(xe(inx)-xc));% k of  line equation

     swit = frameLogical(xc,yc);
     intersection_space = zeros(1,9);
    for  x = xc:steps:xe(inx)
        y =k*(x-(xc))+yc;
        y = round(y);
        values = frameLogical(x,y);
        % fprintf('x=%d y=%d values=%d',x,y, values);
        % disp(values);

           if values ~=swit
                intersection_space(1,count)= x;
                intersection_space(1,count+1)= y;
                intersection_space(1,count+2)= sqrt((xe(inx)-xCenter).^2+(ye(inx)-yCenter)
.^2);
                count = count+3;
                swit = values;
           elseif count>7
               break;
           elseif all(count == 1) && all(x == xe(inx))

                intersection_space(1,1)= xe(inx);
                intersection_space(1,2)= ye(inx);
                intersection_space(1,3)= sqrt((xe(inx)-xCenter).^2+(ye(inx)-yCenter).^2);


          end


    end
    result = [result,intersection_space];





    % drawing result on video
%     test = load('intersection.csv');
%
%     test_matrix = test(:,1:9);
%     test_label = test(:,9);
%     test_matrix = mapminmax('apply',test_matrix', PS);
%     test_matrix = test_matrix';
%     [predict_label_1,accuracy_1,dec_value] = svmpredict(test_label,test_matrix,model,'-b
 1'); % version ,match parameters

%     for indexp = 1:predict_label_1(end)
%     te =predict_label_1(indexp);
%     disp(te);
%     if te ==1
%         result ='walk';
%     elseif te ==2
%         result ='jump';
%     elseif te == 3
%         result ='run';
%     end
%     end
%    %  disp(te);
%     out = insertText(out,[10 10],result);

end

    subplot(2,2,1),imshow(out),title('frame_Box');
    %subplot(2,2,1),imshow(opticalBG),title('OpF');
    hold on ;
    plot(flow, 'DecimationFactor', [2 2], 'ScaleFactor', 20)% FLOW VECTOR
        drawnow
        hold off

%     subplot(2, 2, 2), imshow(frameLogical), title('logicalFrame');
%     hold on
%     plot(yCenter,xCenter, 'r+', 'MarkerSize', 10, 'LineWidth', 3);
%     hold off
%     subplot(2,2,3), imshow(boundary),title('boundaries');
%     hold on
%     plot(yb,xb,'b-','markerSize',10,'lineWidth',2);
%     plot(x,y,'color','r','markerSize',10);
%     hold off
 dlmwrite('intersection.csv',result,'-append');
end
```



Published with MATLAB® R2018a