Lane Departure Warning System

This example shows how to detect road lane markers in a video stream and how to highlight the lane in which the vehicle is driven. This information can be used to detect an unintended departure from the lane and issue a warning.

Introduction

This example detects and tracks road lane markers in a video sequence and notifies the driver if they are moving across a lane. The example illustrates how to use the `HoughTransform`, `HoughLines` and `LocalMaximaFinder` System objects to create line detection and tracking algorithms. The example implements this algorithm using the following steps:

1. Detect lane markers in the current video frame.
2. Match the current lane markers with those detected in the previous video frame.
3. Find the left and right lane markers.
4. Issue a warning message if the vehicle moves across either of the lane markers.

To process low quality video sequences, where lane markers might be difficult to see or are hidden behind objects, the example waits for a lane marker to appear in multiple frames before it considers the marker to be valid. The example uses the same process to decide when to begin to ignore a lane marker.

Initialization

Use these next sections of code to initialize the required variables and System objects.

```
DrawPoly = 1;  % Set to 0 to draw only lines
NumRows = 120; % Number of rows in the image region to process.
MaxLaneNum = 20; % Maximum number of lanes to store in the tracking repository.
ExpLaneNum = 2;  % Maximum number of lanes to find in the current frame.
Rep_ref   = zeros(2, MaxLaneNum); % Stored lines
Count_ref = zeros(1, MaxLaneNum); % Count of each stored line
TrackThreshold = 75; % Maximum allowable change of lane distance
                     % metric between two frames.
LaneColors = single([0 0 0;1 1 0; 1 1 0; 1 1 1;1 1 1]);
% Minimum number of frames a lane must be detected to become a valid lane.
frameFound = 5;
% Maximum number of frames a lane can be missed without marking it invalid.
frameLost = 20;

% For selecting Rho values 35:45 (1-based index: 415:424)
startIdxRho_R = 415;
NumRhos_R = 11;

% For selecting Theta values -90:-70deg (1-based index: 1:21)
startIdxTheta_R = 1;
NumThetas_R = 21;

% For selecting Rho values 379:415 (1-based index: 1:36)
startIdxRho_L = 380;
NumRhos_L = 36;

% For selecting Theta values 55:85deg (1-based index: 146:176)
startIdxTheta_L = 146;
NumThetas_L = 21;

% Offset for displaying the lines
offset = int32([0, NumRows, 0, NumRows]);
```

Create a `VideoFileReader` System object™ to read video from a file.

```
hVideoSrc = vision.VideoFileReader('viplanedeparture.mp4');
```

Create `HoughLines` System objects to find the Cartesian coordinates of the lines defined by the lane markers.

```
hHoughLines1 = vision.HoughLines('SineComputation', 'Trigonometric function');
hHoughLines3 = vision.HoughLines('SineComputation', 'Trigonometric function');
```

define parameters for inserting lane departure warning text.

```
warnText = {sprintf('Right\nDeparture'), '', sprintf(' Left\n Departure')};
warnTextLoc = [120 170;-1 -1; 2 170];
```

define parameters for inserting text specifying lane marker color/type.

```
lineText = {'', ...
        sprintf('Yellow\nBroken'), sprintf('Yellow\nSolid'), ...
        sprintf('White\nBroken'), sprintf('White\nSolid')};
```

Create a `VideoPlayer` System object to display the output video.

```
hVideoOut = vision.VideoPlayer;
```

Initialize the variables used in the stream processing loop.

```
Frame = 0;
NumNormalDriving = 0;
OutMsg = int8(-1);
OutMsgPre = OutMsg;
Broken = false;
```

Stream Processing Loop

Create the processing loop to perform lane detection on the input video. This loop uses the System objects you instantiated.

```
warningTextColors = {[1 0 0], [1 0 0], [0 0 0], [0 0 0]};
while ~isDone(hVideoSrc)
    RGB = step(hVideoSrc);

    % Select the lower portion of input video (confine field of view)
    Imlow  = RGB(NumRows+1:end, :, :);

    % Edge detection and Hough transform
    Imlow = rgb2gray(Imlow); % Convert RGB to intensity
    I = imfilter(Imlow, [-1 0 1], 'replicate','corr');

    % Saturate the values to be between 0 and 1
    I(I < 0) = 0;
    I(I > 1) = 1;

    th = multithresh(I); % compute threshold
    [H, Theta, Rho] = hough(I > th);

    % Convert Theta to radians
    Theta = Theta * pi / 180;

    % Peak detection
    H1 = H;
    % Wipe out H matrix with theta < -78 deg and theta >= 78 deg
    H1(:, 1:12) = 0;
```

```
H1(:, end-12:end) = 0;
Idx1 = houghpeaks(H1, ExpLaneNum, 'NHoodSize', [301 81], 'Threshold', 1);
Count1 = size(Idx1,1);

% Select Rhos and Thetas corresponding to peaks
Line = [Rho(Idx1(:, 1)); Theta(Idx1(:, 2))];
Enable = [ones(1,Count1) zeros(1, ExpLaneNum-Count1)];

% Track a set of lane marking lines
[Rep_ref, Count_ref] = videolanematching(Rep_ref, Count_ref, ...
                            MaxLaneNum, ExpLaneNum, Enable, Line, ...
                            TrackThreshold, frameFound+frameLost);

% Convert lines from Polar to Cartesian space.
Pts = step(hHoughLines1, Rep_ref(2,:), Rep_ref(1,:), Imlow);

% Detect whether there is a left or right lane departure.
[TwoValidLanes, NumNormalDriving, TwoLanes, OutMsg] = ...
        videodepartureWarning(Pts, Imlow, MaxLaneNum, Count_ref, ...
                              NumNormalDriving, OutMsg);
% Meaning of OutMsg: 0 = Right lane departure,
%                    1 = Normal driving, 2 = Left lane departure

% Detect the type and color of lane marker lines
YCbCr  = rgb2ycbcr(double(RGB(NumRows+1:240, :, :)));
ColorAndTypeIdx = videodetectcolorandtype(TwoLanes, YCbCr);
% Meaning of ColorAndTypeIdx:
% INVALID_COLOR_OR_TYPE = int8(0);
% YELLOW_BROKEN = int8(1); YELLOW_SOLID = int8(2);
% WHITE_BROKEN = int8(3);  WHITE_SOLID = int8(4).

% Output
Frame = Frame + 1;
if Frame >= 5
    TwoLanes1 = TwoLanes + [offset; offset]';
    if DrawPoly && TwoValidLanes
        if TwoLanes(4,1) >= 239
            Templ = TwoLanes1(3:4, 1);
        else
            Templ = [0 239]';
        end
        if TwoLanes(4,2) >= 239
            Tempr = TwoLanes1(3:4, 2);
        else
            Tempr = [359 239]';
        end
        Pts_poly = [TwoLanes1(:,1); Templ; Tempr; ...
            TwoLanes1(3:4,2); TwoLanes1(1:2,2)];

        % Draw Polygon for lane
        RGB = insertShape(RGB,'FilledPolygon',Pts_poly.',...
                          'Color',[0 1 1],'Opacity',0.2);
    end

    % Draw lane marker lines
    RGB = insertShape(RGB,'Line',TwoLanes1',...
        'Color',{'yellow','magenta'});
    % Insert Departure warning text (empty text will not be drawn)
    txt = warnText{OutMsg+1};
```

```
        txtLoc = warnTextLoc(OutMsg+1, :);
        txtColor = single(warningTextColors{mod(Frame-1,4)+1});
        RGB = insertText(RGB,txtLoc,txt,'TextColor', txtColor, ...
                         'FontSize',20, 'BoxOpacity', 0);

        % Insert text indicating type and color of left and right lanes
        for ii=1:2
            % empty text will not be drawn
           txtLoc = TwoLanes1([1 2], ii)' + int32([0 -35]);
           lineTxt = lineText{ColorAndTypeIdx(ii)};
           txtColor = LaneColors(ColorAndTypeIdx(ii), :);
           RGB = insertText(RGB,txtLoc,lineTxt,'TextColor',txtColor, ...
                            'FontSize',14, 'BoxOpacity', 0);
        end

        % Draw third lane if needed
        if OutMsgPre ~= OutMsg
            ColorType = ColorAndTypeIdx(2-(OutMsg == 2));
            Broken    = ColorType == 2 || ColorType == 4;
        end
        ShowThirdLane = Broken && (OutMsg~=1);
        if ShowThirdLane
            if OutMsg == 0
                % Find right third lane
                Idx2 = houghpeaks(H(startIdxRho_R:startIdxRho_R+NumRhos_R-1, ...
                          startIdxTheta_R:startIdxTheta_R+NumThetas_R-1), ...
                          'NHoodSize', [7 7], 'Threshold', 1);
                Rhor = Rho(Idx2(:,1) + startIdxRho_R);
                Thetar = Theta(Idx2(:,2) + startIdxTheta_R);
                ThirdLane = step(hHoughLines3, Thetar, Rhor, Imlow);
            else
                % Find left third lane
                Idx3 = houghpeaks(H(startIdxRho_L:startIdxRho_L+NumRhos_L-1 , ...
                          startIdxTheta_L:startIdxTheta_L+NumThetas_L-1),...
                           'NHoodSize', [7 7], 'Threshold', 1);
                Rhol = Rho(Idx3(:,1) + startIdxRho_L);
                Thetal = Theta(Idx3(:,2) + startIdxTheta_L);
                ThirdLane = step(hHoughLines3, Thetal, Rhol, Imlow);
            end

            OutThirdLane = videoexclude3rdlane(ThirdLane, ShowThirdLane,...
                                    TwoLanes, TwoValidLanes, YCbCr);
            OutThirdLane = OutThirdLane(:) + offset(:);
            RGB = insertShape(RGB,'Line',OutThirdLane.','Color','green');
        end
    end
    OutMsgPre = OutMsg;

    step(hVideoOut, RGB);    % Display video
end
```