Hough Transform base lane detection

Charles Khoury

Communication Systems and Network

Motivation and Primer

What is the Hough Transform

- Hough Transform is a feature extraction technique commonly used in image processing.
- Useful or detecting lines, curves and other features that can be defined in a parametric form.
- Transforms the image space into a parameter space where the problem becomes a peak detection problem.
- Unless you want a specific optimization, most of the work is already done for you.
- Pre-processing the image will help get better results.

Why not use Machine Learning ?

- Machine Learning requires a large dataset (in the supervised case).
- Genetic/evolutionary algorithm alongside reinforcement learning models require a lot of computational power.
- Unless tuned carefully it won't perform optimally when applied to edge devices.

Fig1: Lane detection outcome (courtesy of Medium)

Pre-processing

Image Pipeline

- Transform the image to Gray scale
- ROI Cropping
- Perform a homographic transformation (Optional)
- Convert the image to black and white (Optional)
- Perform parametric Hough Transform
- Find peaks
- Get point coordinates

Fig2: Image processing pipeline (Left to Right, Top Down)

Gray Scale Conversion

- Contrast based between the line and the road
- Colour information not needed
- Average Pixel value

$$G_{xy} = \frac{R_{xy} + G_{xy} + B_{xy}}{3}$$

```
I = rgb2gray(RGB)
newmap = rgb2gray(map)
```

Fig3: Gray Scale Conversion

Region of Interest

- Not all the image contains useful information.
- In our case only the bottom half of the image contains useful data.
- Of that bottom the road ahead is of interest.
- Perform Element-wise matrix multiplication with a mask.

$$I = \begin{pmatrix} G_{11} & \cdots & G_{1N} \\ \vdots & \ddots & \vdots \\ G_{N1} & \cdots & G_{NN} \end{pmatrix}, \mathrm{M} = \begin{pmatrix} 0 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix}$$

```
function C = imask(img)
    figure
    imshow(img)
    mask = drawpolygon;
    C = createMask(mask);
    close
end
grayRoad = grayRoad.*C;
```

Fig4: ROI cropping

Homographic Transform

- Get a top-down view of the scene.
- Linear Transformation.
- With a 2D image this is perform using a 3x3 matrix.
- Note the extra dimension.
- Inverse transform of H matrix to dewarp.

$$\boldsymbol{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}, \boldsymbol{v} = \begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix}, \boldsymbol{v}' = \begin{bmatrix} v'_x \\ v'_y \\ 1 \end{bmatrix}$$

$$\boldsymbol{v}' = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix}$$

$$\boldsymbol{v} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}^{-1} \begin{bmatrix} v'_x \\ v'_y \\ 1 \end{bmatrix}$$

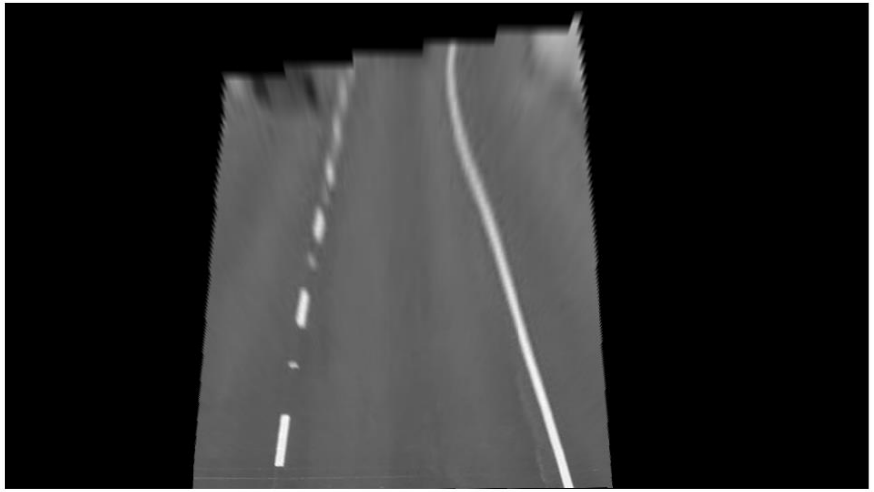


Fig5: Perspective warping

```
function H = iwarp(img)
    figure
    imshow(img)
    sow = drawpolygon; %get orignal transform point
    dow = drawpolygon; %define the new points
    origin = sow.Position;
    mapped = dow.Position; map_coor = mapped;
    H = fitgeotrans(origin,map_coor, 'projective');
    close
end
```

Hough Transform - Theory

- In the cartesian plane: (a, b)
- Parametrise in the polar coordinate using (ρ, θ)
- $\frac{\rho}{b} = \sin(\theta) \Leftrightarrow \frac{1}{b} = \frac{\sin(\theta)}{\rho}$
- $\frac{\rho}{b} a = \cos(\theta) \Leftrightarrow \frac{a}{b} = \frac{\cos(\theta)}{\rho}$
- $\rho = x\cos(\theta) + y\sin(\theta)$
- MATLAB function returns those parameters and the parameter space matrix H (row = ρ, columns = θ).
- Each pair represents a bin, crossing implies a more “active” bin.

```
[H,theta,rho] = hough(edge(img,'canny'));
```

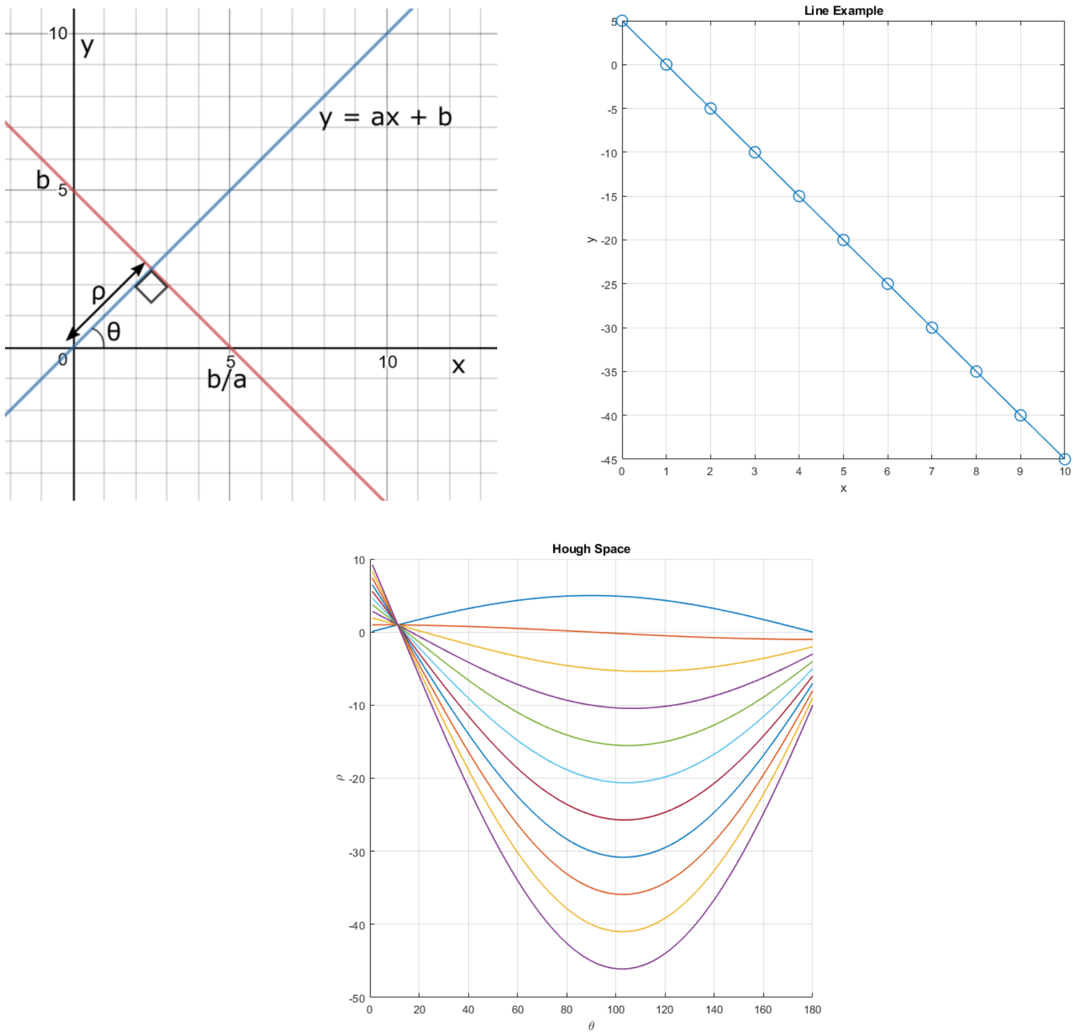


Fig6: ROI Hough Theory

Hough Transform - Example

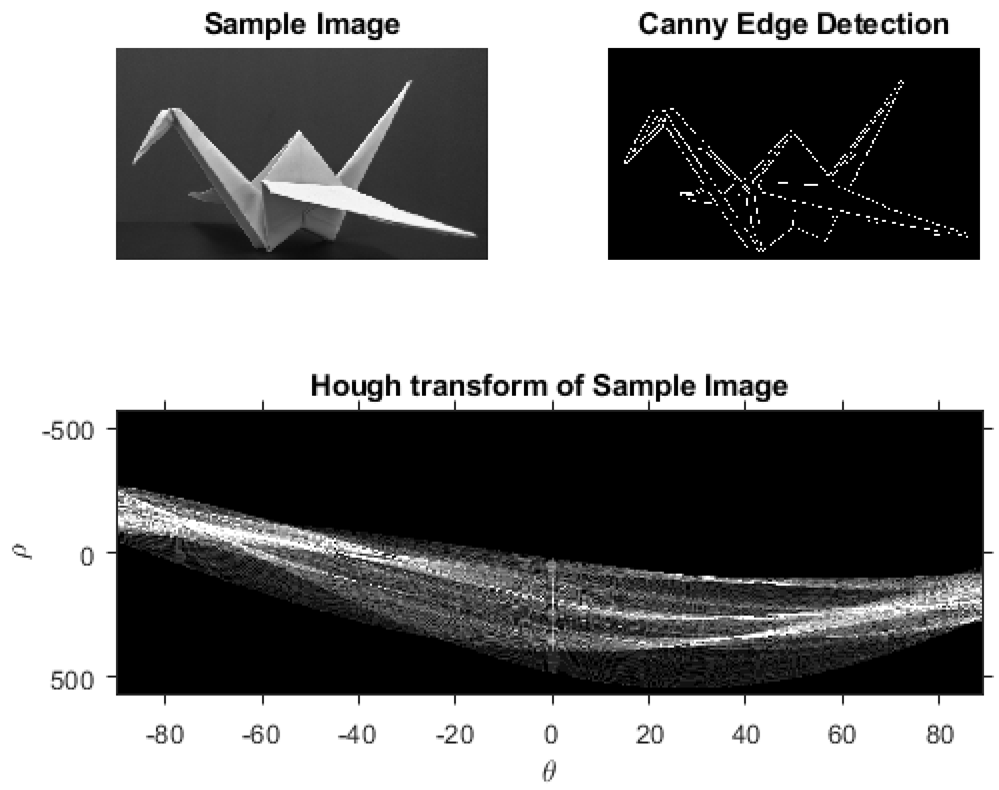


Fig7: Hough Transform example

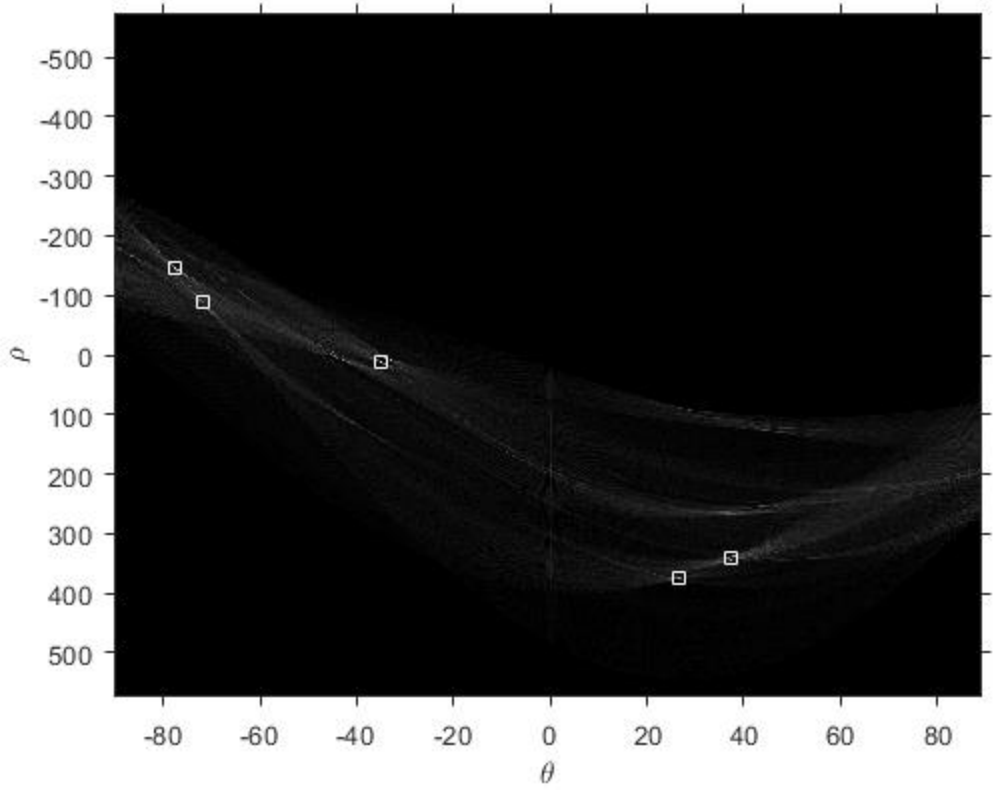
Hough Transform - Binning

- The bins are encoded in the H matrix
- Find the peaks using MATLAB's peak function
- Peak coordinates in vector form can then be converted to (ρ, θ)
- Use the MATLAB's houghline method to return (a, b) coordinates

Fig8: Lane detection

```
P = houghpeaks(H,20,'threshold',ceil(0.3*max(H(:))));
x = theta(P(:,2)); %% (matlab list comprehension)
y = rho(P(:,1));
lines =
houghlines(img,theta,rho,P,'FillGap',30,'MinLength',5);
for k = 1:length(lines)
        %%% show lines
end
```

Hough Transform – Example continued



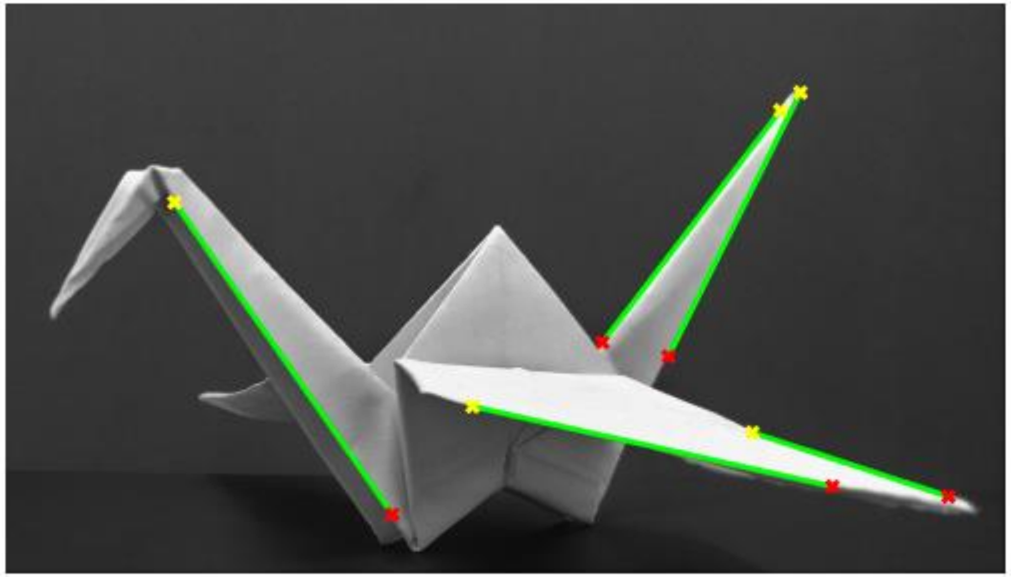


Fig9: Hough Transform example continued

Conclusion

- Most of the work has already been done for you !!
- You are piecing together different algorithms to achieve your desired outcome.
- An understanding of the algorithms remains crucial.
- You may wish to improve/change a certain aspect of the image processing pipeline to improve performance, accuracy, etc.
- Tutorial is available on Github.

You Only Look Once (YOLO)

- Deep Learning algorithm for object recognition
- Output dimension/target vector:
  - ROI grid size
  - Anchors
  - Number of classes
  - Bounding box positional and size
- Steps:
  - Segment the image in ROI grid
  - For each segment of the image retrieve target vector.
  - Apply non-max suppression:
    - Get bounding boxes for each element of the grid.
    - Get rid of low probability predictions.
    - Independently run non-max suppression for each class
- As the name implies it is extremely efficient (for an ML model of that size/depth). The image is fed once and information on all classes are extracted

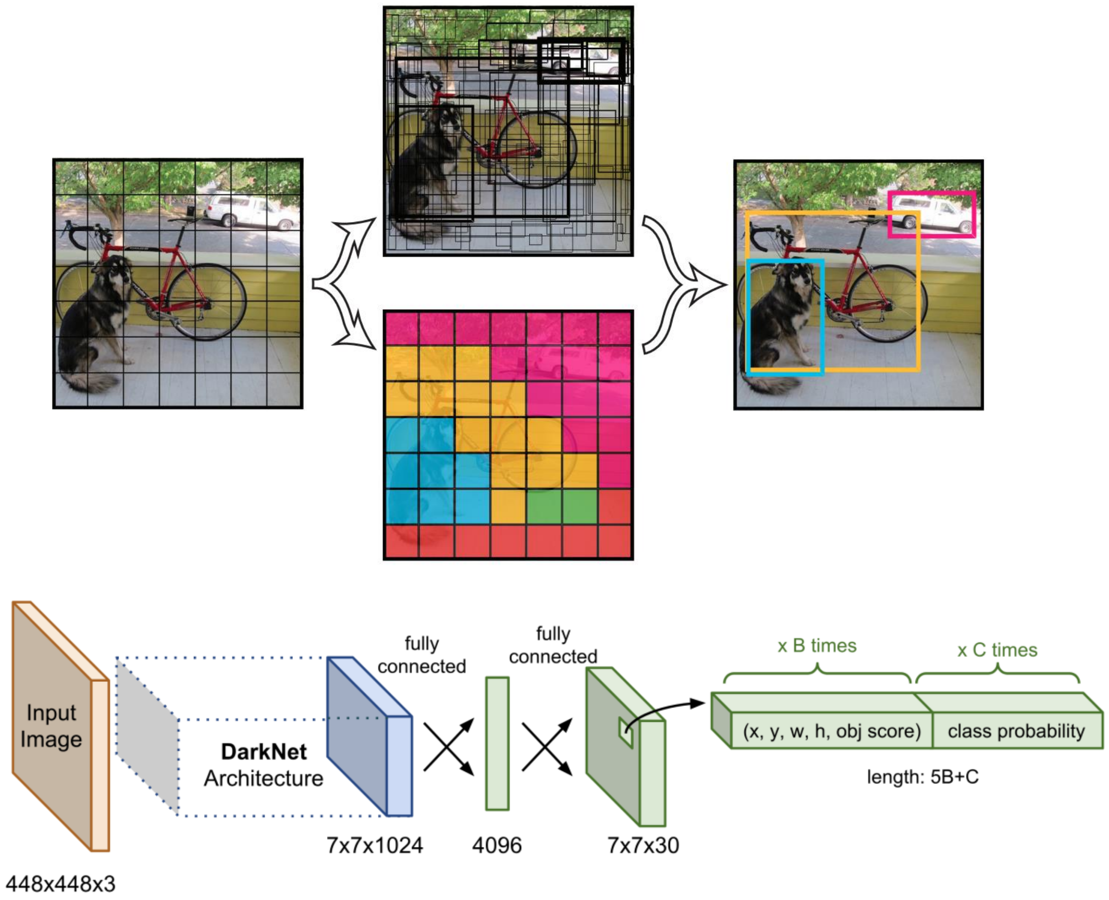


Fig10: YOLO framework