Practice 1: Finding lane lines with Hough transform

Carlos Villaseor

***Abstract*—In this Report we present the process to create a progam able to find the lines araund a lane out of a video recorded by a camera installed in a car. First is developed the segmentation module and the hough transform on an image then develop the complite program to be used with recorded videos.**

***Keywords*—comuter vision, Hough transform.**

I. Introduction

In computer vision is comon the problem were we need to find the geometry of an object in an image. One way is to find the lines in the image, and for this we use the Hough transform. we can use this metod to find a path for a vehicle or find an area where a robot has to move. This tecnic is named after its original inventor Paul Hough, in where the edges vote for a posible line location, each edge point votes for all posible lines passing though it, and the lines with more votes are examined to be a potential line.

II. preprocessing

in order to process de image first we have to made a preprocessing, this stage is used to preapare the image reducing the noice and focusing on espesific parts of the image, this is made to avoid problems with the prossesing or simplifying the operations needed to get the results.

A. Gauss blurr

First we use a gaussian blur to reduce noice in the image. openCV has a metod wse can use, this metod smooth the image so the noise we have in the image blend with the pixels near it, so this is done by convolving each point in the input array with a Gaussian kernel and then summing them all to produce the output array. the Gaussian kernel is generated by the Gaussian equation shown in equation 1

$$G_0(x,y) = A\exp(\frac{-(x-\mu_x)^2}{2\sigma_x^2} + \frac{-(y-\mu_y)^2}{2\sigma_y^2}) \tag{1}$$

the metod used in openCV is show in listing 1

Listing 1. Gaussian blur

```
GaussianBlur(src ,dst ,Size(i ,i ),0 ,0);
```

B. Canny filter

The next sep in the preprocessing is the Canny filter, this filter let us find the contours (edge detector), this metod retun a image with the points where the value of a pixel have changed. the metod follw this steps:

1) A convolution masks in x and y direction using the kernels (equation 2).

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{2}$$

2) Then find the gradient strength and direction with the equations 3 and 4

$$G = \sqrt{G_x^2 + G_y^2} \tag{3}$$

$$\theta = \arctan(\frac{G_y}{G_x}) \tag{4}$$

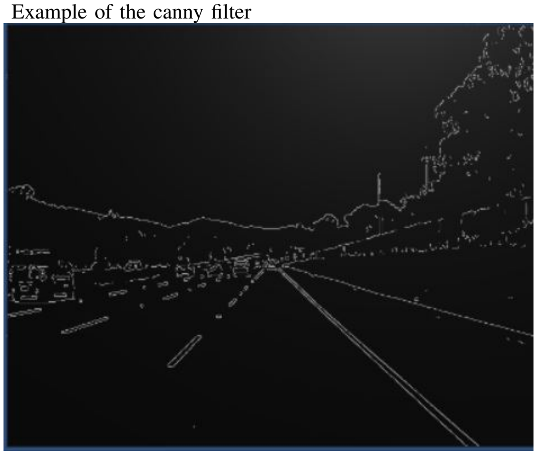
To simplify the proses we are going to use the image ina gray sacle insted of a RGBA, then aply the canny filter, the metod is used as shown in listing 2 where imgMat is the original image and grayMat is the image in gray sacle.

Listing 2. Canny filter

```
Imgproc.cvtColor(imgMat, grayMat,
    Imgproc.COLOR_RGB2GRAY);
Imgproc.Canny(grayMat, grayMat,
    uimbral1, uimbral2);
```

an exaple of the canny filter is show in figure 1

Fig. 1. Example of the canny filter



C. Mask

As a final step before aply the hough transform, we aply a mask. this mask is show in figure 2. this will errase thi data we dont use, because the problem is to detect tje lines in a line of a video which is monted iftont of a car and the focus point of the inage is near the center of the image (see figure 3, we only need the data inside the triange made from the center and the corners in the bottom (see listing 3).

Fig. 2. mask

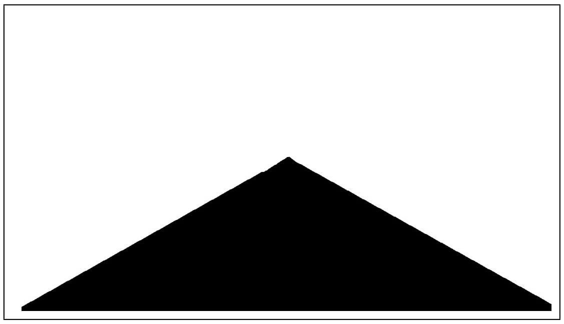
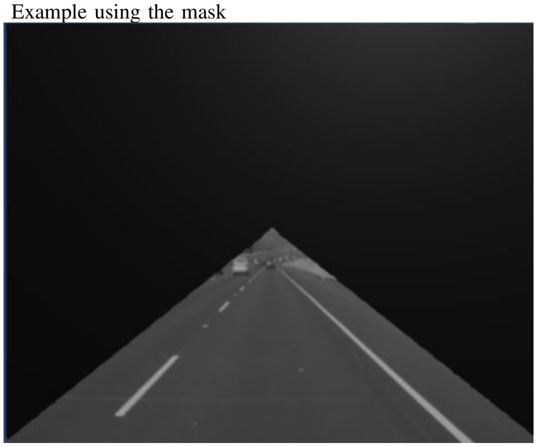


Fig. 3. Example using the mask



Listing 3. Mask

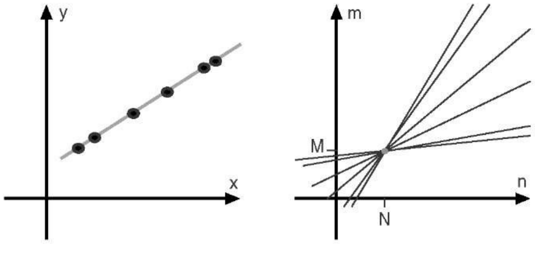
```
grayMat = grayMat - maskMat;
```

III. Hough transforms

The hough tansform uses the posible points of a line and maps them using ρ and θ isnted of using x and y (see equation 5), the equation to descride a line in hough space is represented as a single point, and a point in hough space is represented as a line (see figure 4).

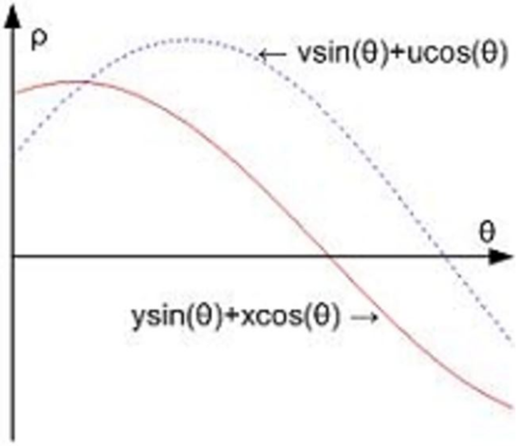
$$y = \theta * x + \rho \tag{5}$$

Fig. 4. At left is a line and some ponits from the line and, at rigth is their representation in hough space



Normaly the lines in hough space are represented in function of they parameters sin and cos (see figure5) because when we plot this way we get a periodic fuction and make shure that the points will intersect somewere and do not have to gess where in the space they will intersect.

Fig. 5. At left is a line and some ponits from the line and, at rigth is their representation in hough space

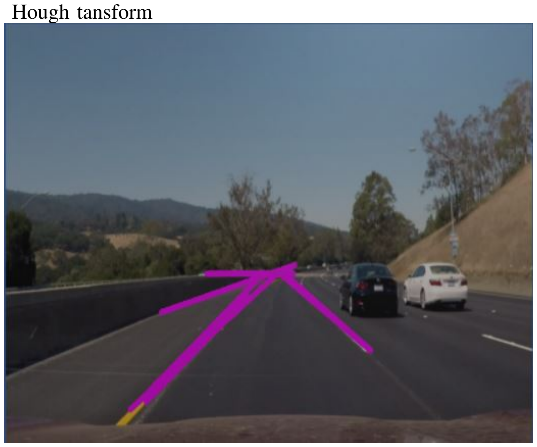


After that all we need to do is find where are more intersections, this proses is caled to "vote", each intersection give a vote for the line represented in the hough space by the intersection point. we can use the metod as show in listing 4 this give the output show in figure 6

Listing 4. Hough Transform

```
Imgproc.HoughLinesP(grayMat, lines, rho,
    Mathf.PI / 180, houghVotes,
    minLineLength, maxLineGap);
```

Fig. 6. Hough tansform

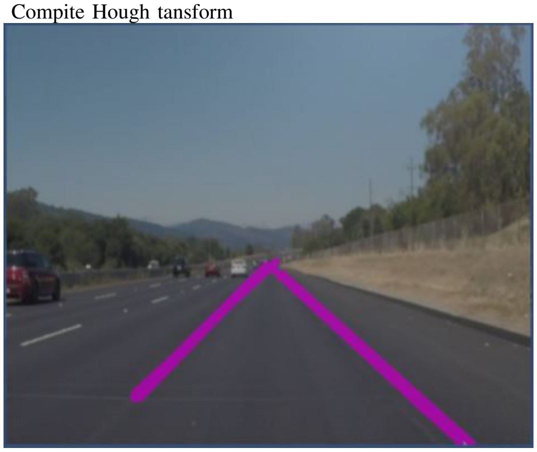


Now we have a lot of lines obtained by the transformation, we have to select just the important ones, we can select using the tilt of the line, but cheking the lines we want are always the biger ones so using a theshold we can find what we want usinf the code in listing 5, we have in linesArray all the lines separated in two points,initial pount and end point, and each point in x and y, and aux is the value in pixels of the heigt of the lines we not want then paited in purple. The out put is show in figure 7

Listing 5. Threshold Transform

```
for(int i=0;i<linesArray.Length;i=i+4)
{
  if((linesArray[i+1]-linesArray[i+3]<-aux||
      linesArray[i+1]-linesArray[i+3]>aux))
  {
    Imgproc.line(auxMat,
                 new Point(linesArray[i],
                 linesArray[i+1]),
                 new Point(linesArray[i+2],
                           linesArray[i+3]),
                 new Scalar(255,0,255),
                 10);
  }
}
```

Fig. 7. Compite Hough tansform



IV. Conclusion

inconclutioin using the hough transformation can be used in may ways in which the data of the line is needed, but we have to be carful because is easy to get wrong data or not enough and the proses will not get the expected results.

References

[1] Richard Szeliski, *Computer Vision: Algorithms and Applications*, 1st ed. Springer, 2010.

[2] Open Source Computer Vision Library, http://opencv.org/, 2017.