Line Following with Six Wheel Robot

Takahiro Miki

September 2, 2016

1 Black line detection

In this section, a line detection algorithm is considered. The objective is to recognize where is the black line in the image like Figure 1. The flow of detecting

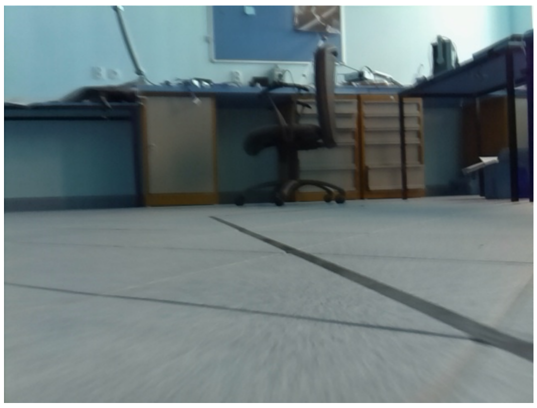

Figure 1: Objective is to detect the black line

black line is like below.

1. extract black part
2. apply filter to detect edge
3. calculate the horizon
4. apply hough transform to detect line

1.1 Extracting black part

The images has three RGB values. Each represents how bright in Red, Green and Blue. The black color's value of R, B and G seems to be equal. Therefore, black color can be detected with the distance between RGB. Let's define the distance as below.

$$d = \sqrt{(R-G)^2 + (R-B)^2 + (G-B)^2} \tag{1}$$

By using this distance, a black_ratio is defined as below.

$$black\_ratio = \left(1 - \frac{d}{411}\right)^4 \tag{2}$$

where, 411 is the max distance and multiplying four times makes black_ratio be more smaller when d is small. Four times is determined by some experiments.

Using this black ratio, a filtered gray scale image can be calculated with the equation below.

$$grayscale = 0.3R + 0.59G + 0.11B \tag{3}$$

$$pixel = (255 - grayscale) \times black\_ratio \tag{4}$$

Lastly, binarize the image with certain threshold.

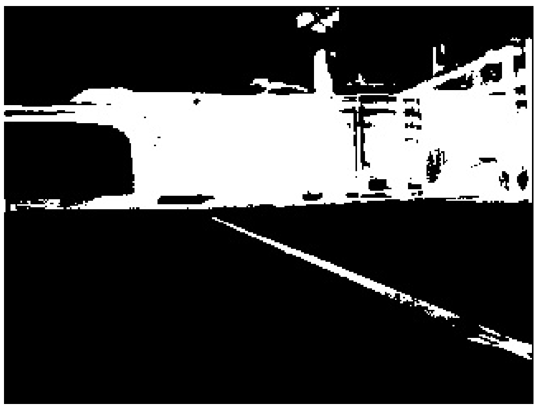

Figure 2: After extracting black part

1.2 Applying Laplacian Filter

By applying laplacian filter, the edge of the black part appears.

1.3 Detect horizon

After applying the filter, we want to use Hough transform to detect the line. But, the problem is there are many lines above the horizon line. So, before applying the hough transform, detecting the horizon line is needed. To detect the horizon, we calculated the variance along the x axis. In each rows, calculate

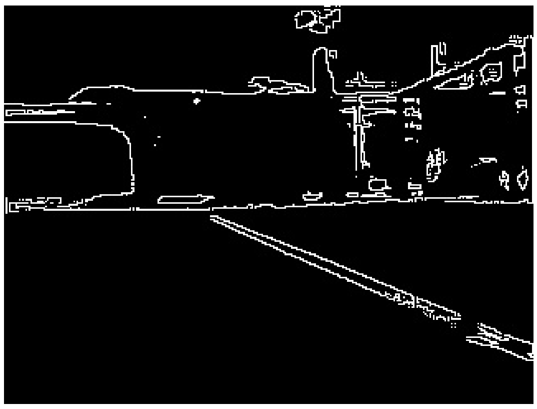

Figure 3: After applying laplacian filter

how the black edge is distributed. If the value is high, it should not be a line and there should be above the horizon.

So, scan from bottom and if the variance is bigger that a certain threshold, that line should be a horizon.

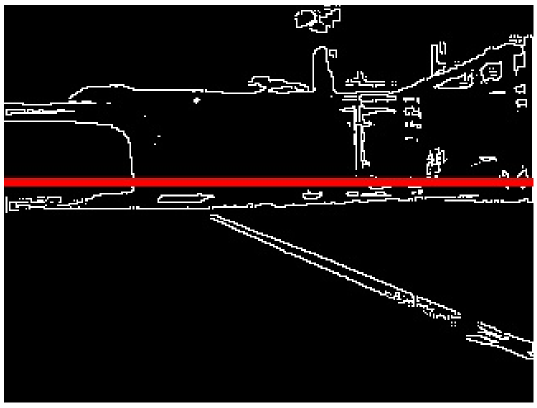

Figure 4: After detecting horizon

1.4 Apply Hough transform

After detecting the horizon, cut the image and use only below the horizon. The result of applying hough transform to the cutted image is Figure1.4

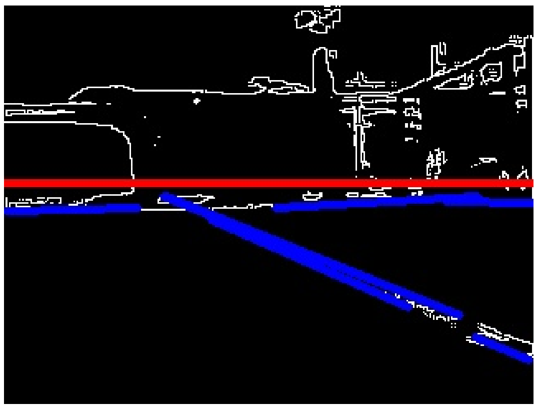

Figure 5: After applying hough transform

2 Calculating look ahead point

To use the result of line detection, the problem is that the image is captured from the vehicle camera and the position on the image is different from the position in the real world. To solve this problem, transformation from the vehicle viewpoint image to the top view image is used.

2.1 Homography transform

The Homography transformation is a popular geo-referencing technique used worldwide. It is based on quite complex geometric and mathematic concepts, known as "homogeneous coordinates" and "projective planes", the explanation of which is not within the scope of this document.

The transformation is based on this formulation,

$$u = \frac{ax + by + cz}{gx + hy + 1} \tag{5}$$

$$v = \frac{dx + ey + fz}{gx + hy + 1} \tag{6}$$

This formulation can be described with matrix as below,

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \tag{7}$$

Like in [2], I calculated the homography matrix by defining four points transformation(Figure 2.1). The result is like below.

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix} = \begin{pmatrix} 0 & 22.2 & 0 \\ -0.75 & 7.33 & 480.0 \\ 0 & 0.03 & 1 \end{pmatrix} \tag{8}$$

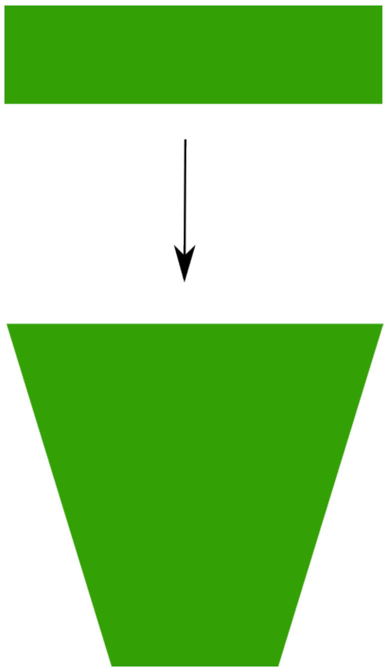
Figure 6: Definition of the four points transformation. Calculate the homography matrix to transform from upper square to the lower square

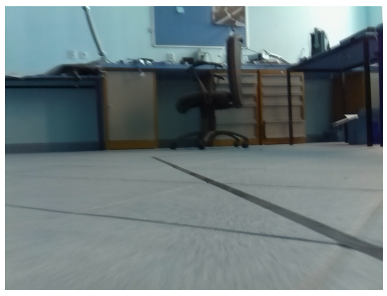
Figure 7: Before transformed

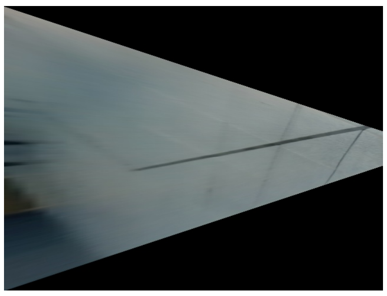
Figure 8: After transformed

Then, to reduce the computation, this homography transformation is applied only for detected lines. After this transformation, we have several lines in top view perspective.

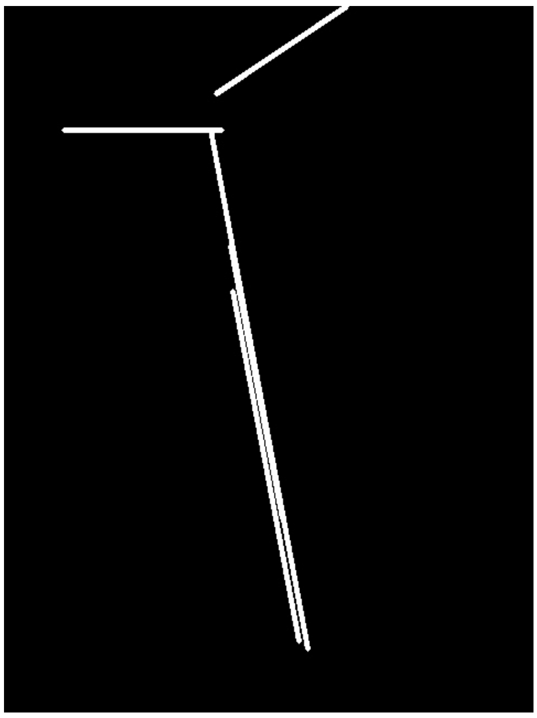

Figure 9: Transforming only the lines.

2.2 Calculating the intersect point with lines and a circle

To calculate the look ahead point, we need to calculate the intersect point with detected lines and a circle. Each lines are consist with two points and a circle's center is $(0, 0)$, radius is r.

Let the first point $A(x_0, y_0)$ and the second point $B(x_1, y_1)$ and define the vector as $\vec{A} = (-x_0, -y_0)$, $\vec{B} = (-x_1, -y_1)$ and $\vec{S} = \vec{A} - \vec{B}$.

2.2.1 Detecting the line is crossed with the circle

Before calculating the intersection points, we detected if the line is crossed with the circle. There are two patterns of crossing. The pattern with one point is inside, other is outside, and the pattern with both points are outside the circle.

With the pattern 1, if one point is inside and the other point is outside the

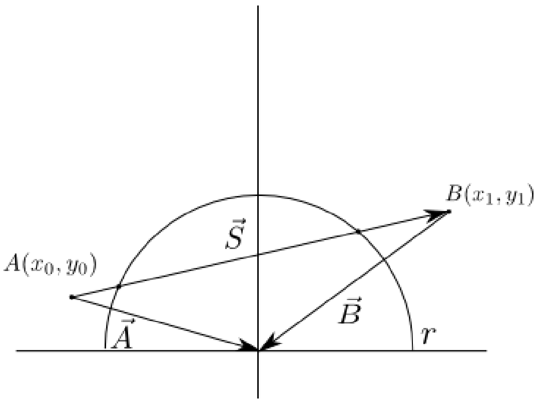

Figure 10: Definition used in intersection point calculation.

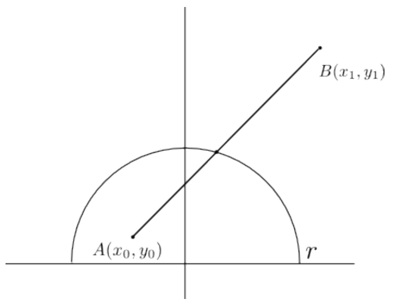

Figure 11: Pattern1 one point is inside and the other is outside the circle

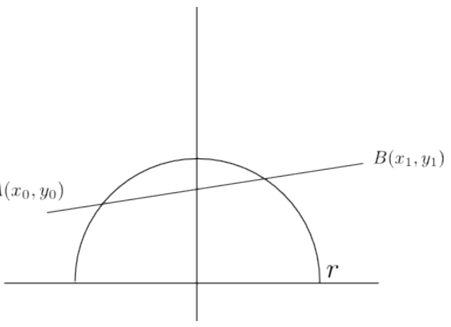

Figure 12: Pattern2 both points are outside the circle

circle, the line is detected to be crossed.

$$\sqrt{x_0^2 + y_0^2} < r \wedge \sqrt{x_1^2 + y_1^2} > r \tag{9}$$

or,

$$\sqrt{x_0^2 + y_0^2} > r \wedge \sqrt{x_1^2 + y_1^2} < r \tag{10}$$

With the pattern 2, if the closed distance of the line is smaller than r, the line seems to be crossed.   Distance d can be calculated as below.

$$\begin{aligned} d &= |\vec{A}| \sin\theta && (11) \\ \vec{A} \times \vec{S} &= |\vec{A}| \cdot |\vec{S}| \cdot \sin\theta && (12) \end{aligned}$$

$\therefore$

$$d = (\vec{A} \times \vec{S}) / |\vec{S}| \tag{13}$$

But, there are some cases although d is smaller than r but, the line is not crossed. To get rid of this case, if the two angle is bigger than 90°, the line is

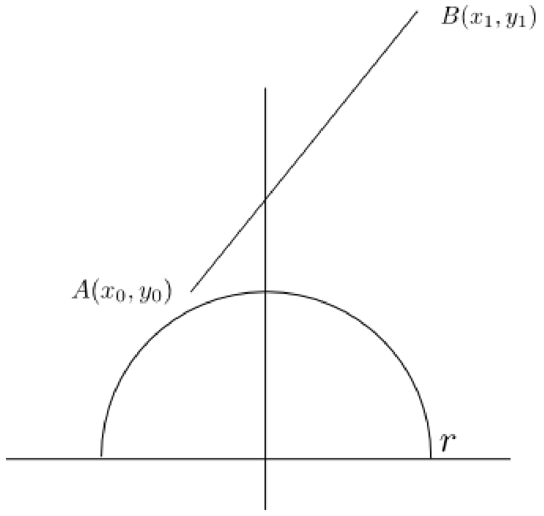

Figure 13: The line does not cross though distance is smaller that r.

detected to be not crossed. By using inner product,

$$\begin{aligned} \vec{A} \cdot \vec{S} &= |\vec{A}| \cdot |\vec{S}| \cdot \cos\theta_A && (14) \\ \vec{B} \cdot \vec{S} &= |\vec{B}| \cdot |\vec{S}| \cdot \cos\theta_B && (15) \end{aligned}$$

if $(\vec{A} \cdot \vec{S}) \cdot (\vec{B} \cdot \vec{S}) > 0$, the line is not crossed.

So, the line is crossed when,

$$|\vec{A}| < r \wedge |\vec{B}| > r \tag{16}$$

or,

$$|\vec{A}| > r \wedge |\vec{B}| < r \tag{17}$$

or,

$$(\vec{A} \times \vec{S})/|\vec{S}| < r \wedge (\vec{A} \cdot \vec{S}) \cdot (\vec{B} \cdot \vec{S}) < 0 \tag{18}$$

2.2.2 Intersection point calculation

After detecting whether the line is crossed with a circle, the intersection point can be calculated with a equation of circle and line.

Line

$$y = \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) + y_0 \tag{19}$$

$$y = ax + b \tag{20}$$

where,

$$a := \frac{y_1 - y_0}{x_1 - x_0} \tag{21}$$

$$b := y_0 - \frac{y_1 - y_0}{x_1 - x_0} x_0 \tag{22}$$

Circle

$$x^2 + y^2 = r^2 \tag{23}$$

From these equations, intersection points are

$$x^2 + (ax + b)^2 = r^2 \tag{24}$$

$$x = \frac{-ab \pm \sqrt{a^2 b^2 - (1 + a^2)(b^2 - r^2)}}{1 + a^2} \tag{25}$$

Use the points which x is between x_0 and x_1.

If $x_0 = x_1$,

$$x = x_0 = x_1 \tag{26}$$

$$y = \sqrt{r^2 - x^2} \tag{27}$$

2.2.3 Look ahead point calculation

From the line detection algorithm, there are several lines. Here, we have to choose which intersection point to use and how to decide the radius.

First, the radius is started from a certain value given by hand. If there are intersection points, use these values. If there is not, use a smaller radius and calculate the intersection points again. Continue this process until an intersection point is detected. If there is no intersection point until r becomes 0, it means that there is no line.

Second, with several intersection points, use the point that is most closest to the center.

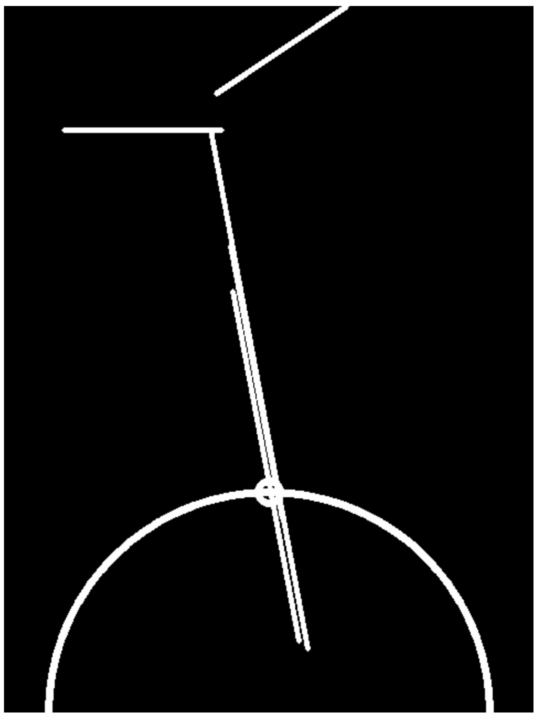

Figure 14: The result of calculating the look ahead point

3 Line Following Control with pure pursuit algorithm

3.1 Pure Pursuit

Pure pursuit is the algorithm to follow the path by setting a goal point and calculate the curvature to move towards that point[1].

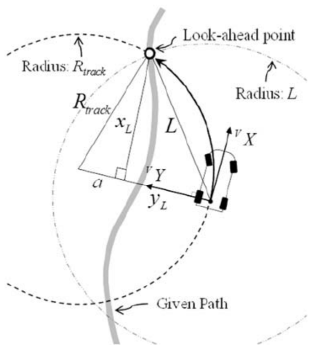

Figure 15: Geometry of the pure pursuit

3.2 Robot Model

To go to the goal point, the vehicle has to turn in a circular path. Here, the vehicle is modelled to be a two wheel car.

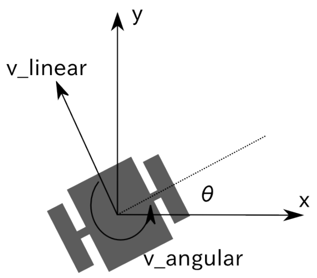

Figure 16: Model of the robot

The velocity of the robot can be described as below.

$$v_x(t) = -v_l \sin\theta(t) \quad (28)$$
$$v_y(t) = v_l \cos\theta(t) \quad (29)$$

where, v_l is the linear velocity of vehicle and v_x and v_y are the velocity in x axis and y axis.

The angle θ can be calculated by using an angular velocity of the vehicle v_a.

$$\begin{aligned}\dot{\theta}(t) &= v_a &(30)\\ \theta(t) &= v_a t + \theta_0 &(31)\\ &= v_a t &(32)\end{aligned}$$

where, θ_0 is assumed to be 0.

Using this equation, the position of the robot can be calculated. Velocity should be

$$\begin{aligned}v_x(t) &= -v_l \sin(v_a t) &(33)\\ v_y(t) &= v_l \cos(v_a t) &(34)\end{aligned}$$

and, the position is

$$x(t) = \frac{v_l}{v_a}\cos(v_a t) + x_0 \qquad (35)$$

$$y(t) = \frac{v_l}{v_a}\sin(v_a t) + y_0 \qquad (36)$$

By assuming that $x(0), y(0) = 0$, $x_0 = -\frac{v_l}{v_a}, y_0 = 0$

So, the equation of circular path would be like below.

$$\left(x + \frac{v_l}{v_a}\right)^2 + y^2 = \left(\frac{v_l}{v_a}\right)^2 \qquad (37)$$

This is a circle which center is $\left(\frac{v_l}{v_a}, 0\right)$ and radius is

$$r = \frac{v_l}{v_a} \qquad (38)$$

(Figure 3.2)

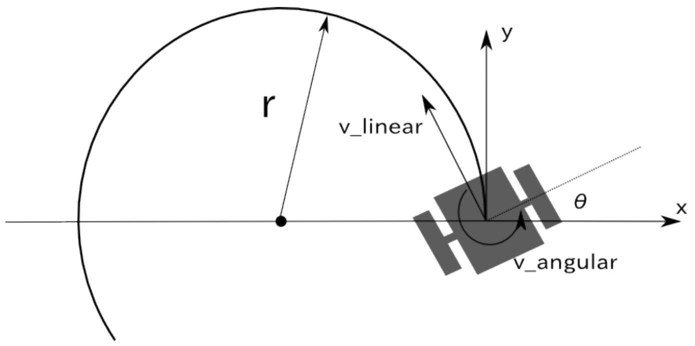

Figure 17: Circular path of robot

3.3 Calculation of the control velocity

When the look ahead point is given, the linear velocity v_l and angular velocity v_a can be calculated. Set (x_g, y_g) to be the look ahead point.

Then required radius r can be calculated as below.

$$(x_g + r)^2 + y_g^2 = r^2 \tag{39}$$

$$r = -\frac{x_g^2 + y_g^2}{2x_g} \tag{40}$$

From Equation(38), v_l and v_a can be calculated respectively. By assuming that v_l is a constant value, v_a is

$$v_a = -\frac{2x_g v_l}{x_g^2 + y_g^2} \tag{41}$$

To avoid too big v_a, a maximum value of angular velocity is defined.(v_a^*) If calculated v_a is bigger than v_a^*, v_l and v_a can be calculated as below.

$$v_a = v_a^* \tag{42}$$

$$v_l = -\frac{x_g^2 + y_g^2}{2x_g} v_a \tag{43}$$

3.4 Transforming control velocity to each motor command

The linear velocity and angular velocity are determined by the speed of each wheel, the length of shaft and the radius of the wheel. If there are rotary encoders on the wheels, measuring each values and decide the motor commands is effective. However, in the platform now we use does not have encoders so the motor commands are calculated by some parameters. These parameters are estimated by doing some experiments.

We used three parameters *linear_v_ratio*, *angular_v_ratio* and *left_right_ratio*. *linear_v_ratio* and *angular_v_ratio* determines the left and right wheel command. *left_right_ratio* adjust the difference between left wheel's and right wheel's speed when the robot does not go straight with $v_a = 0$. Each motor's command can be determined like below.

$$left\_motor\_command = (v_l \times linear\_v\_ratio + v_a \times angular\_v\_ratio) \times left\_right\_ratio \tag{44}$$

$$right\_motor\_command = v_l \times linear\_v\_ratio - v_a \times angular\_v\_ratio \tag{45}$$

References

[1] Coulter, R. C. Implementation of the pure pursuit path tracking algorithm. Tech. rep., DTIC Document, 1992.

[2] 鳥屋剛毅, 北原格, AND 大田友一. K-006 位置情報付き俯瞰画像を用いたモバイルカメラの位置と方位の推定 (マルチメディア・仮想環境基礎, k 分野: 教育工学・福祉工学・マルチメディア応用). **情報科学技術フォーラム講演論文集** *11*, 3 (2012), 541–546.