Xenobot technical report
Lane following

Shengwen Cheng , Po-Sheng Chen

January 2017

1 Introduction

Xenobot is a computer vision based self-driving system inspired by MIT duckietown project. We re-implement our own system for real-time robotics research and may add more new features in the future.
The algorithm we're using is a modified version of MIT duckietown, so you can find many similarities between two projects. This report is focus on how our lane following algorithm works.

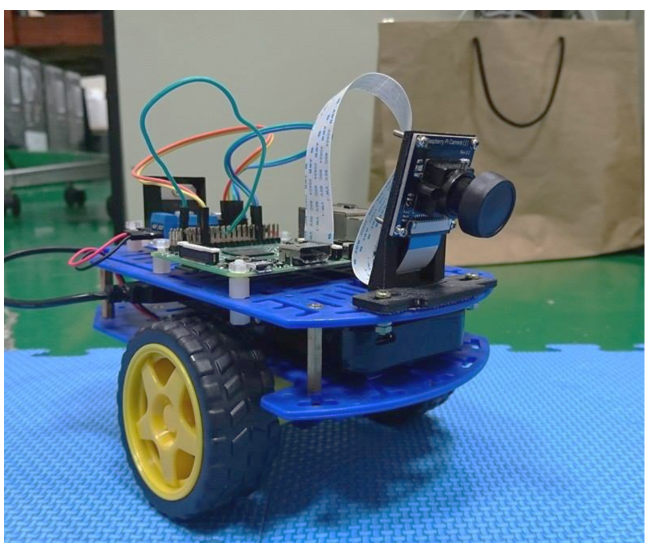

Figure 1: Xenobot

2 Camera calibrations

Camera calibration is a big issue for computer vision. The camera parameter tells us the mapping between the image and the real world.

2.1 Intrinsic parameters

Intrinsic parameter describe the mapping between 2D image frame and 3D camera frame, usually also dealing with the distortion or skew of the camera.

2.2 Extrinsic parameters

Extrinsic parameters describe the rotation and translation of the camera with respect to the world frame.

3 Lane pose estimation

3.1 Lane pose and system conventions

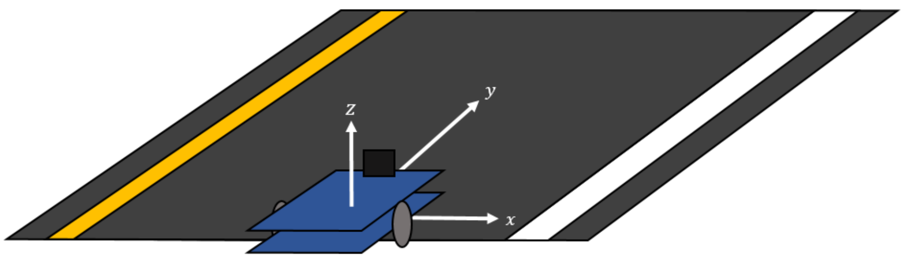

Figure 2: Lane coordination

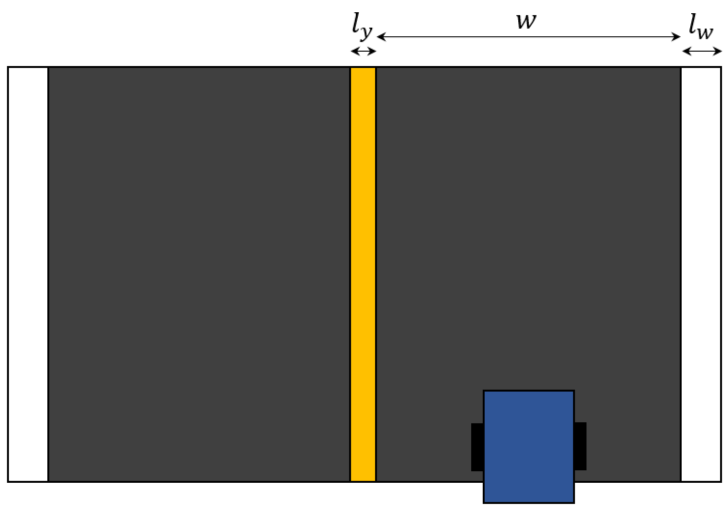

Figure 3: 2D view of lane

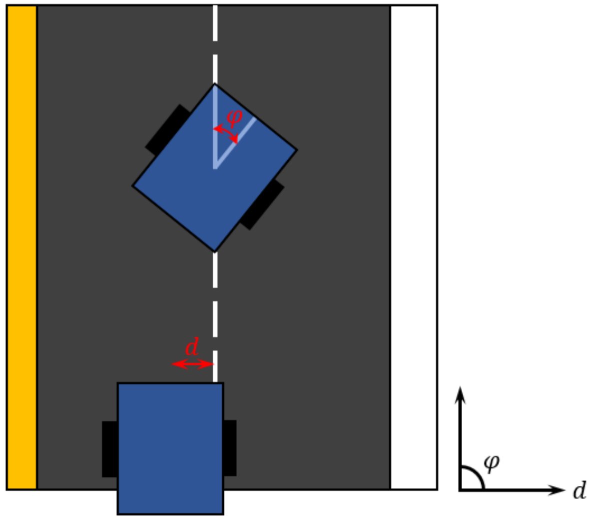

Figure 4: Lane pose

3.2 Segements detection

The first step for lane pose estimation is to extract the segments from the image, the process to do this is to threshold the specific color we want, and apply canny edge detector and Hough transform so we can find out the location of those segments

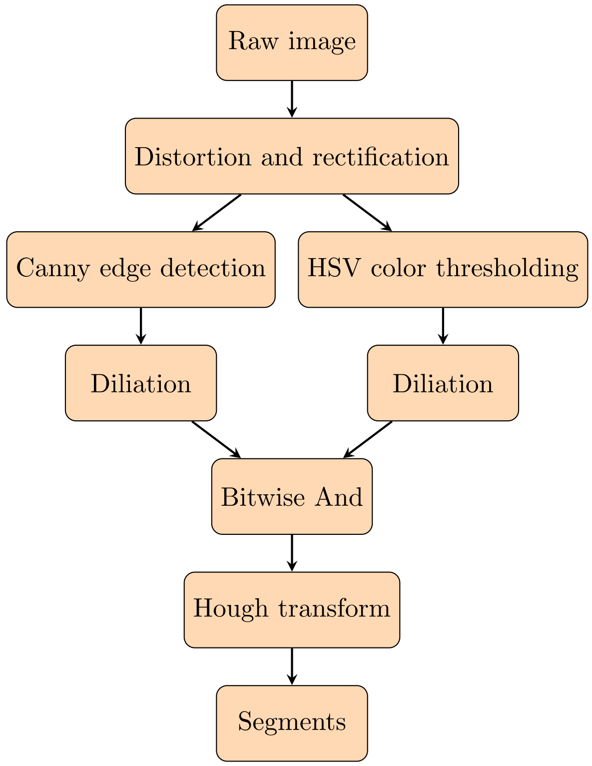

Figure 5: Work flow of lane detector

3.3 Frame transformation

The segments we got in the lane detector stage is in the camera frame, however, what we are really interested is the car frame, so we need to do a transformation between these two frame.

$$\begin{pmatrix} x_{car} \\ y_{car} \end{pmatrix} = \begin{pmatrix} x_{camera} - r \cdot \sin\phi \\ y_{camera} - r \cdot \cos\phi \end{pmatrix}$$

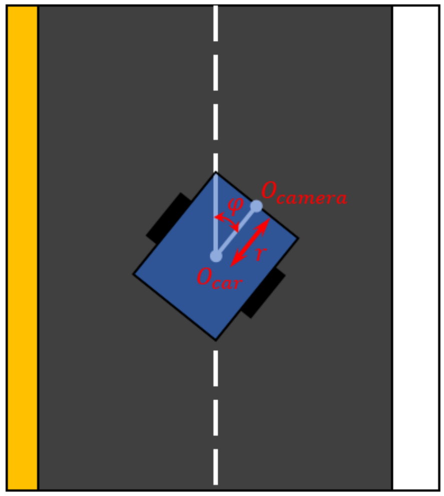

Figure 6: Frame transformation

3.4 Segment side recognition

We obtained the lane segments during the lane detection process. Next step is to figure out the segment side on the lane mark. It could be determined by reading multiple pixel values in the direction of segment normal vector on color thresholding image.

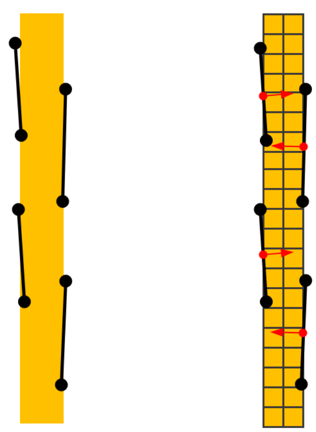

Figure 7: Lane segments

Algorithm 1: Segment side recognition

Data: segment, accumulator threshold, color binarization image
Result: side (left or right)
$\vec{P_1} = (x_1, y_1)$
$\vec{P_2} = (x_2, y_2)$
$\vec{P} = (\vec{P_1} + \vec{P_2})/2$
$\vec{t} = \frac{\vec{P_2} - \vec{P_1}}{\left\|\vec{P_2} - \vec{P_1}\right\|}$
$\vec{n} = (-y_t, x_t)$
for $i < pixel\ count$ **do**
    $x \leftarrow \lceil x_p + x_n \cdot i \rceil$
    $y \leftarrow \lceil y_p + y_n \cdot i \rceil$
    if $I(x, y) = I_{max}$ **then**
        $left \leftarrow left + 1$
    $x \leftarrow \lfloor x_p - x_n \cdot i \rfloor$
    $y \leftarrow \lfloor y_p - y_n \cdot i \rfloor$
    if $I(x, y) = I_{max}$ **then**
        $right \leftarrow right + 1$
end
if $left > threshold$ & $right < threshold$ **then**
    return is left
else if $right > threshold$ & $left < threshold$ **then**
    return is right
else
    return unknown side

3.5 Segment pose estimation

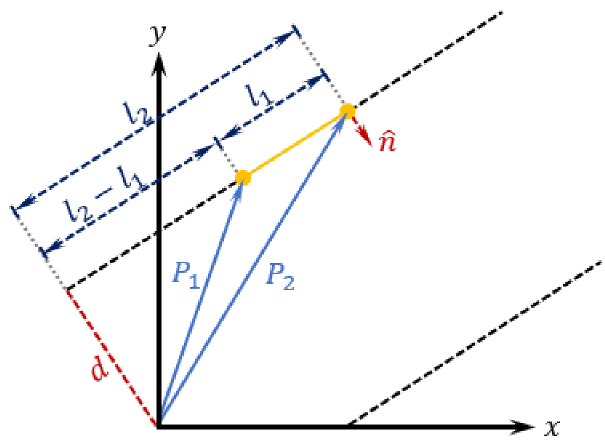

Figure 8: Lane geometry

Algorithm 2: Generate vote

Data: segment
Result: pose d_i and ϕ_i

$\vec{P_1} = (x_1, y_1)$
$\vec{P_2} = (x_2, y_2)$
$\vec{t} = \frac{\vec{P_2} - \vec{P_1}}{\left\| \vec{P_2} - \vec{P_1} \right\|}$
$\vec{n} = (-y_t, x_t)$
$\phi_i = \arctan(\frac{y_t}{x_t}) - \pi/2$
if *segment color = white* **then**
    if *edge side = right* **then**
        $\vec{k} = (\frac{w}{2} + l_w) \cdot \vec{n}$
    else
        $\vec{k} = (\frac{w}{2}) \cdot \vec{n}$
    end
else if *segment color = yellow* **then**
    if *edge side = left* **then**
        $\vec{k} = (-\frac{w}{2} - l_y) \cdot \vec{n}$
    else
        $\vec{k} = (-\frac{w}{2}) \cdot \vec{n}$
    end
$\vec{j} = (r \cdot \sin \phi, r \cdot \cos \phi)$
$\vec{P_1'} = \vec{P_1} + \vec{k} - \vec{j}$
$\vec{P_2'} = \vec{P_2} + \vec{k} - \vec{j}$
$d_1 = \vec{P_1} \cdot \vec{n}$
$d_2 = \vec{P_2} \cdot \vec{n}$
$d_i = (d_1 + d_2)/2$

Figure 9: Vote generation

3.6 Histogram filter

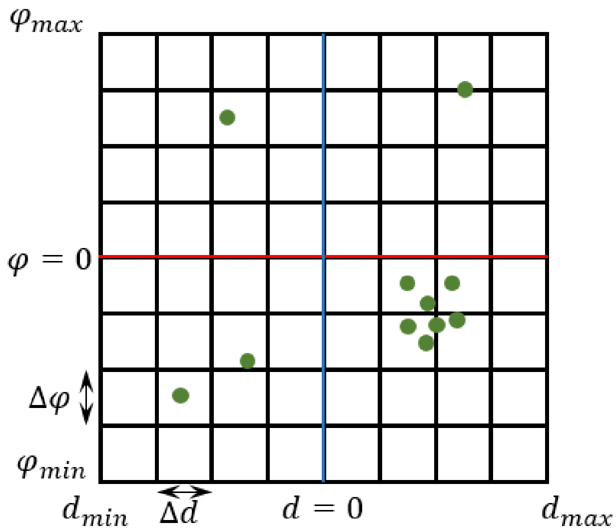

Figure 10: Histogram filter

Algorithm 3: Histogram filter

Data: segment

Result: filtered pose d and ϕ

for ***all*** *segments* **do**

$\quad (\phi_i, d_i) \leftarrow generate\_vote(segment)$

$\quad I \leftarrow round(\frac{\phi_i - \phi_{min}}{\Delta\phi})$

$\quad J \leftarrow round(\frac{d_i - d_{min}}{\Delta d})$

$\quad histogram(I, J) \mathrel{+}= 1$

end

$(I_{highest}, J_{highest}) \leftarrow find\_highest\_vote()$

$\phi_{histogram} \leftarrow I_{highest} \cdot \Delta\phi + \phi_{min}$

$d_{histogram} \leftarrow J_{highest} \cdot \Delta d + d_{min}$

$\phi_{mean} \leftarrow 0\ ,\ d_{mean} \leftarrow 0$

$N_\phi \leftarrow 0\ ,\ N_d \leftarrow 0$

for ***all*** (ϕ_i, d_i) **do**

$\quad$ **if** $\phi_i \in [\phi_{histogram} - \frac{\Delta\phi}{2}, \phi_{histogram} + \frac{\Delta\phi}{2}]$ **then**

$\quad\quad \phi_{mean} \mathrel{+}= \phi_i$

$\quad\quad N_\phi \mathrel{+}= 1$

$\quad$ **if** $d_i \in [d_{histogram} - \frac{\Delta d}{2}, d_{histogram} + \frac{\Delta d}{2}]$ **then**

$\quad\quad d_{mean} \mathrel{+}= d_i$

$\quad\quad N_d \mathrel{+}= 1$

end

$\phi_{mean} \leftarrow \frac{\phi_{mean}}{N_\phi}$

$d_{mean} \leftarrow \frac{d_{mean}}{N_d}$

Figure 11: Histogram filter

4 Control system

4.1 Differential wheels

4.2 PID Controller

The equation of PID controller in continuous time is given as:

$$e(t) = setpoint(t) - x(t)$$

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}$$

in discreted time:

$$e[t] = setpoint[t] - x[t]$$

$$u(t) = K_p e[t] + K_i \sum_0^t e[t] \Delta t + K_d \frac{e[t] - e[t-1]}{\Delta t}$$

4.3 d control and phi control

The lane pose controller of Xenobot is a cascaded PID controller, the phi controller is treat as a low level contoller for lane orientation stablizing, the higher level d controller will change the setpoint of the phi controller to turn left or right back to the middle of road when it is needed.

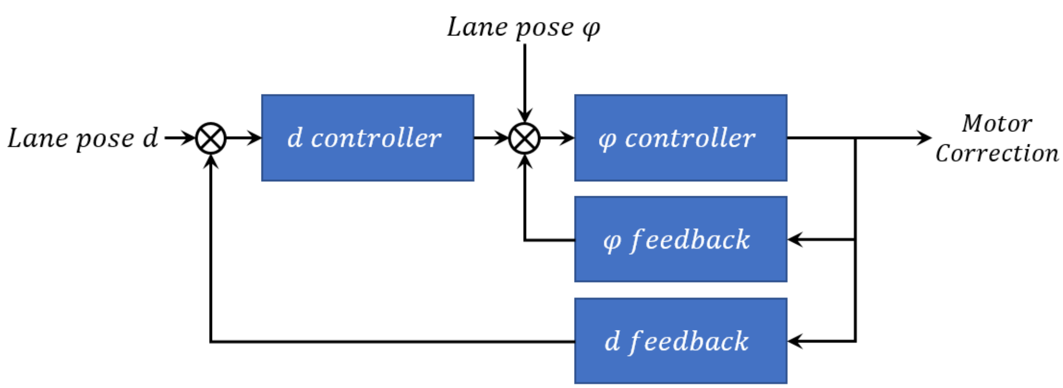

Figure 12: Control diagram

5 References

[1] Lane Filter by Liam Paull, MIT CSAIL