Lane Filter

Liam Paull

Overview: The purpose of this document is to describe the lane filter module that filters the individual line segment detections and produces an estimate of the position and angle of the car within the lane. The general approach is similar to a histogram filter. Each detected segment uniquely defines a position and orientation of the robot within the lane. Therefore, each segment gets a "vote" that is used to build a measurement likelihood function for each incoming image. The posterior belief of the position and angle of the car in the lane is then determined using a Bayes' Filter. The mode of distribution (MAP estimate) is reported as the current position and orientation or if the distribution has entropy above a threshold then it is reported that no consensus was found (indicating that we aren't sure what's going on probably better to transfer control back to human).

Assumptions: There is an assumption that the lane colors on right and left are different (currently assuming left=yellow, right=white). However, there is no assumption that the vehicle is actually *in* the lane. A visual depiciton of the basic setup is shown in Fig. 1.

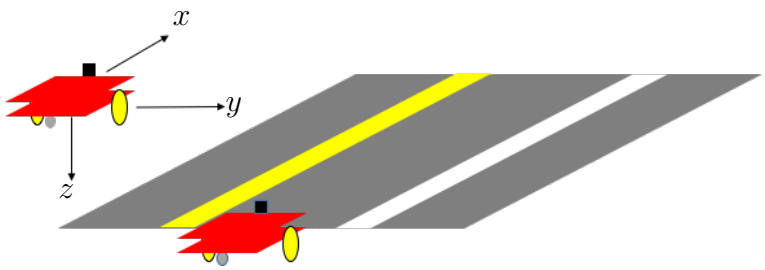

Figure 1: The car looking at a lane

Input: The lane detectors receives a list of N segments we will refer to

them here as $s^{1:N}$ and the corresponding ROS msg is given by:

```
duckietown_msgs/Segment[] segments
uint8 WHITE=0
uint8 YELLOW=1
uint8 RED=2
uint8 color
```

where each segment, s^i is specified by two points p_1^i, p_2^i (either in pixel space or in the body frame of the robot):

```
duckietown_msgs/Pixel[2] pixels
geometry_msgs/Point[2] points
```

Here we are assuming that the segments have been pre-rectified and treat them as if they are points on the 2D plane as shown in Fig. 2. The convention is that *as you travel from point 1 to point 2 along a segment, the colored lane should be on your right hand side*. Each point is on the ground plane and therefore specified in $\mathcal{R}^2$: $p_1^i \triangleq [x_1^i, y_1^i]$ where we use the convention of

Output: The output should be a lane reading ROS msg of the form:

```
float32 d   #lateral offset
float32 sigma_d
float32 phi #heading error
float32 sigma_phi
int32 status
#Enum for status
int32 NORMAL=0
int32 ERROR=1
```

1 Proposed Method

The filtering method proposed is a direct application of the Bayes' filter. Let's review first:

1.1 Background - The Bayes' Filter

Material taken mostly from [**?**]. Define the posterior belief about state x_t as:

$$bel(x_t) \triangleq p(x_t|z_{1:t}, u_{1:t}, x_0) \tag{1}$$

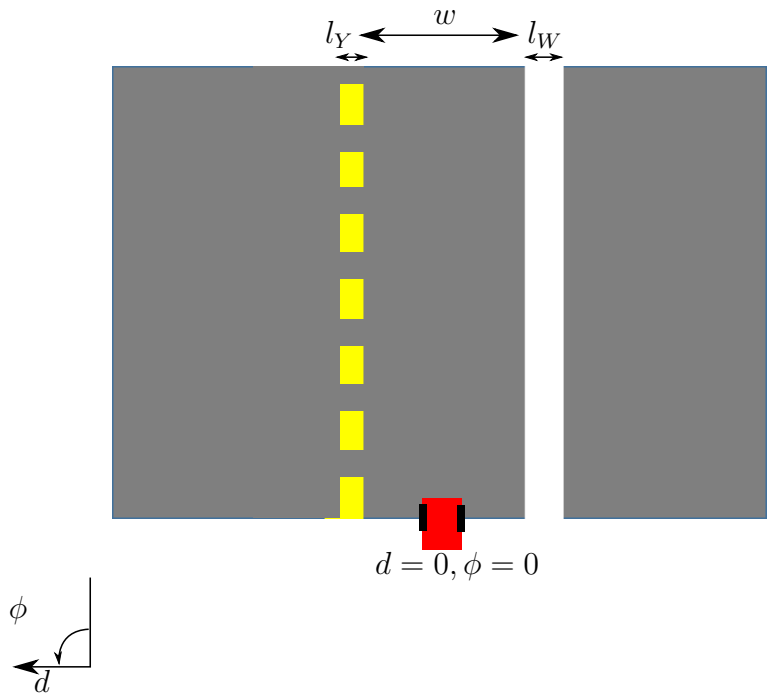

Figure 2: Top-down view of the car in the lane

The posterior belief is updated recursively using the following two step predict-update cycle:

$$\begin{aligned} \bar{bel}(x_t) &= \int p(x_t|x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1} \\ bel(x_t) &= \nu p(z_t|x_t) \bar{bel}(x_t) \end{aligned} \tag{2}$$

which is repeated over the entire domain of x_t. The probability density function (pdf) $p(x_t|x_{t-1}, u_t)$ is the process model and dictates how to use the *control inputs* and the previous state to get to the current state. The pdf $p(z_t|x_t)$ is the measurement model and specifies the measurement likelihood - or the probability that we would get measurement z_t given that we are at state x_t. The value of ν is such that the result is a valid pdf that sums to 1 (it is a normalization factor that accounts for $p(z_t)$ in Bayes rule).

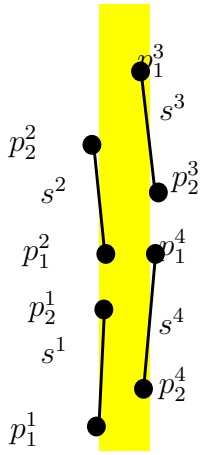

Figure 3: A lane with segments indicated that are the output of the line detection module.d

1.2 Preliminaries

The state of the car in the lane at time t should be represented by the tuple: $x_t \triangleq [d_t, \phi_t]^T$, where d_t is the lateral displacement of the car in the lane (with the $d = 0$ line being defined as the center of the lane) and the ϕ_t is the angle relative to the center axis. The width of the lane should be defined as w and the widths of the right (white) and left (yellow) lanes will be defined as l_W and l_Y.

The control u_t is represented by estimates linear and angular velocity (v_t and ω_t respectively) of the vehicle: $u_t = [v_t, \omega_t]^T$.

1.3 The Process Model

The process model in this case is fairly straightforward and is largely determined by geometry:

$$\begin{aligned} d_t &= d_{t-1} + \Delta t v_t \sin \phi_{t-1} + \eta_1 \\ \phi_t &= \phi_t + \Delta t \omega_t + \eta_2 \end{aligned} \tag{3}$$

where $\eta_1 \sim \mathcal{N}(0, \sigma_1)$ and $\eta_2 \sim \mathcal{N}(0, \sigma_2)$ are normally distributed random variables.

Three options are available for dealing with the control inputs:

1. Assume they are zero (zero-velocity) model and hope the noise terms can account for the vehicle motion
2. Infer the linear velocity, v_t, only - either from some sensor or from a mapping from control inputs
3. Infer the linear and angular velocities - either from some sensor of from a mapping from control inputs

The other issue is how to select the noise covariances, σ_1 and σ_2.
If we define the zero-noise process model in vector form:

$$f(x_{t-1}, u_t) = \begin{bmatrix} d_{t-1} + \Delta t v_t \sin \phi_{t-1} \\ \phi_t + \Delta t \omega_t \end{bmatrix} \tag{4}$$

and stack the covariances into matrix: $\Sigma = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}$ then the final probabilistic process model can be compactly written as:

$$p(x_t | x_{t-1}, u_t) = \mathcal{N}\left(x_t; f(x_{t-1}, u_t), \Sigma\right) \tag{5}$$

which is applied in (2).

1.4 The Measurement Model

Each incoming **list of segments** constitutes a single measurement. Each segment allows us to uniquely estimate the lateral displacement and relative angle of the vehicle in the lane. However, it is expected that there will be many outliers. Therefore processing each segment as an individual measurement is not desirable since these outliers are likely to cause rapid divergence of the estimate. Instead, we propose that each segment produces a "vote" where these votes are binned in a histogram and the entire histrogram represents the measurement likelihood. This has the additional advantage that it is non-paramentric (assumes nothing about the distribution of the pdf) and can therefore handle multi-modal and non-Gaussian distributions.

The process to obtain a single vote from a segment is outlined in Algorithm 1 and is based on geometry.

The process in Algorithm 1 is repeated for each incoming segment. The results are added to a histogram as shown in Fig. 4. Once all of the segments have been processed, the histogram is normalized and then used as the pdf to perform the measurement update.

Algorithm 1 Generate_Vote

Input: A single segment in the body frame s^i

Output: The corresponding position and angle in the lane indicated by the segment: d^i,ϕ^i.

$d^i = 0.5(x_1^i + x_2^i)$
if $s^i.color == WHITE$ **then**
  $d^i -= w/2$
  if $x_2^i > x_1^i$ **then**
    $d^i -= l_W$
  end if
else
  $d^i += w/2$
  if $x_2^i > x_1^i$ **then**
    $d^i += l_Y$
  end if
end if
$\phi_i = \pi/2 - \text{atan2}\left(\frac{|x_2^i - x_1^i|}{y_2^i - y_1^i}\right)$

1.5 Applying the Bayes' Filter

Now we synthesize eveything to produce the proposed lane filter. An overview is given in Alg 2.

The filter requires some prior to begin: $bel(x_0)$. In initial development, this should be a zero mean Gaussian with low covariance (assuming that you place the car correctly in the lane at the start of the test). Subsequently, in the case of transitioning from an open-loop behavior to a lane following behavior, this could be transmitted from the "intersection localizer".

The main loop itereates over every state in the domain $dom(x)$ which is defined as:

$$dom(x) = d_{min} : \Delta d : d_{max}, \phi_{min} : \Delta\phi : \phi_{max} \tag{6}$$

where Δd and $\Delta\phi$ are parameters that specify the bin size in the histogram as shown in Fig. 4.

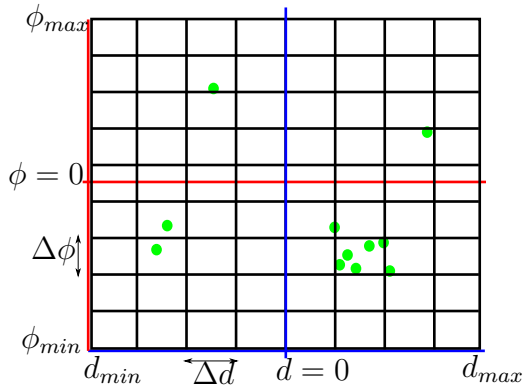

Figure 4: The measurement likelihood as a result of processing one list of segments. Each green dot corresponds to a vote generated by an individual segment.

Algorithm 2 Lane_Filter

$bel(x_0) \leftarrow \mathcal{N}(x_0; 0, \Sigma_0)$
$t \leftarrow 0$
while In Lane Following Mode **do**
  Read/Infer control values u_t
  Read incoming list of segments $s^{1:N}$
  for all Segment s^i **do**
    $(d^i, \phi^i) \leftarrow$ Generate_Vote(s^i)
    Add (d^i, ϕ^i) to histogram
  end for
  Normalize histogram to generate $p(z_t|x_t)$
  for all $x_t \in dom(x_t)$ **do**
    Generate $p(x_t|x_{t-1}, u_t)$ from (4) and (5)
    $\bar{bel}(x_t) \leftarrow \sum_{dom(x_{t-1})} p(x_t|x_{t-1}, u_t) bel(x_{t-1})$
    $bel(x_t) \leftarrow \nu p(z_t|x_t) \bar{bel}(x_t)$
  end for
  $t \leftarrow t + 1$
end while

1.6 Entropy Threshold

The lane filter should also output a status. This status is generated by evaluating the Shannon entropy of the posterior belief:

$$H(bel(x)) = -E[\log(bel(x)] \tag{7}$$

if the posterior entropy is higher (high entropy is equivalent to high uncertainty) than a threshold H_{min} then the status of the filter will be reported as "ERROR". This threshold should be used to trigger the transition from and open loop behavior to a lane following behavior. Additionally, if an "ERROR" status is reported during lane following mode then control of the vehicle should transfer back to the joystick.

1.7 Summary of Parameters

Symbol	**Meaning**
σ_1, σ_1	The noise covariances in the process model.
d_{min}, d_{max}	The minimum and maximum displacement from the center of the line. Empirically this shoud constitute the error in the conroller beyond which we cannot recover
ϕ_{min}, ϕ_{max}	The minimum and maximum anglular offset in the lane beyond which we cannot recover.
$\Delta d, \Delta\phi$	The bin sizes in the d and ϕ directions
H_{min}	The minimum allowable posterior entropy to consider the output of the lane filter to be valid.

Table 1: Summary of Paraneters Used in Lane Filter