Traverse Points

Yuhao Zhang

April 15, 2018

1 Introduction

This project is a model of the car-like robot and a control algorithm to traverse several waypoints with specific coordinates and poses. The code is for Sunfounder's PiCar-V platform and no sensors are involved: the car receives no feedback from its motion and surroundings.

2 Kinematics and control law

Ackerman model can be used to describe the kinematics of a car-like robot:

$$\frac{dx}{dt} = v\cos\theta,$$

$$\frac{dy}{dt} = v\cos\theta,$$

$$\frac{d\theta}{dt} = \frac{v}{L}\tan\gamma,$$

where (x, y) is the position of the middle point of the back wheel axis of the robot in world reference frame. θ is the angle of pose of the robot. The steering wheel angle is γ and the velocity of the back wheel is v. L is the length of the vehicle or wheel base.

These equations can be re-written as:

$$\begin{pmatrix} \frac{dx}{dt} \\ \frac{d\omega}{dt} \\ \frac{d\theta}{dt} \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix}.$$

With the initial position-pose (x, y, θ) and the goal (x^*, y^*, θ^*), it is more convenient to write the equations in polar system via a transformation:

$$\rho = \sqrt{\Delta_x^2 + \Delta_y^2},$$

$$\alpha = \arctan \frac{\Delta_y}{\Delta_x} - \theta,$$
$$\beta = -\theta - \alpha + \theta^*.$$

The linear control law for $-\pi/2 < \alpha \leq \pi/2$, i.e. the waypoint is in front of the vehicle is:

$$v = k_\rho \rho,$$
$$\omega = k_\alpha \alpha + k_\beta \beta,$$

where k_ρ, k_α, k_β are arbitrary coefficients that satisfies $k_\rho > 0, k_\beta < 0, k_\alpha - k_\rho > 0$.

The control law for the cases where the waypoint is behind the vehicle is the same as above, but with transformed angles:

$$\alpha' = -\pi - \beta,$$
$$\beta' = -\pi - \alpha,$$

and $v' = -v$.

3 Pose estimation

Without the feedback from sensors, it is required to estimate the motion from the kinematics of the robot directly. Using the linear approximation of the kinematics equations for a short time period Δt one can obtain

$$\Delta(x, y, \theta) = (v \cos\theta \Delta t, v \sin\theta \Delta t, v/L \tan\gamma \Delta t).$$

Alternatively, the exact solution can be obtained by solving these differential equations directly:

$$\Delta(x, y, \theta) = (R_b \sin(K\Delta t), R_b(1 - \cos(K\Delta t)), K\Delta t),$$

where $R_b = L/\tan\gamma$ and $K = v/R_b$. However, this estimation is non-linear and makes the control law purposed unstable. Therefore the linear approximation will be used in the following sections.

4 Control algorithm implementation

The control law purposed in Sec.(2) and the pose estimation algorithm purposed in Sec.(3) have been implemented in `drive_to_points.ipynb`. Given the input `waypoints.txt`, the generated trajectory and the estimated poses along it is plotted as Fig.()

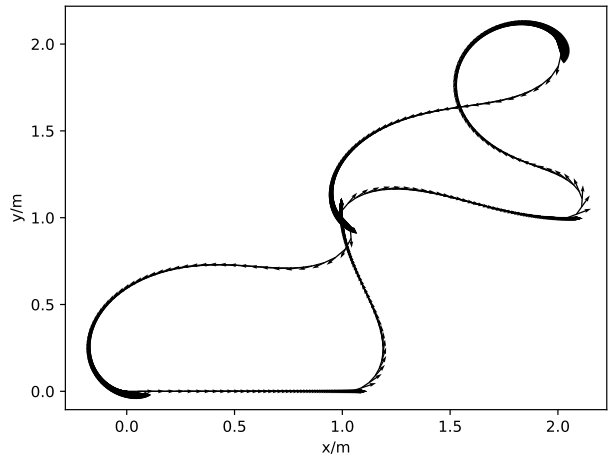

Figure 1: 1000 different 100×100 lattices with different p values, the shortest path and the life time of fire

(a) Beginning with Brown corpus and removing the stop-words and punctuation, making everything lowercase.

(b) Selecting the most commonly used words in the corpus. V : most commonly used 5000 words and C : most commonly used 1000 words.

(c) For each word $w \in V$ and denote context B_w^k: in a text stream, for a word w, we denote the surrounding $2k$ words of every occurrence of w as its context, with k words before it and k words after it.

(d) Define $Pr(c|w)$ as given $w \in V$, the probability that $c \in C$ and $c \in B_w^k$.

(e) Then calculate

$$\Phi_c(w) = max(0, \log \frac{Pr(c|w)}{Pr(c)})$$

for each w.

(f) Apply PCA to $\Phi_c(w)$, to acquire a 100-dimensional representation of $\Phi_c(w)$, denoted as $\Psi(w)$

5 Nearest neighbor test

The result of nearest neighbor search for a random collection of 25 words and the corresponding cosine similarity is listed in Tab.(1). There are pairs such as *proceedings, midst*, easy to correspond to sentences like *...in the midst of proceedings....* And pairs such as *external, internal*, *letter, book*, *day, week* and *gray, green*, which are words in the same domain. There

are also confusing pairs such as *jail, di*. What is more, it also shows some worrying result like *customers, drugs*

Table 1: Experiment results for nearest neighbor

w	c	Cosine similarity
external	internal	0.463563
committed	straightened	0.228246
disposal	specialists	0.252870
commission	department	0.473976
chair	knees	0.473293
torn	lap	0.197125
judges	stem	0.214150
joe	cousin	0.460737
proceedings	midst	0.053407
customers	drugs	0.344528
proved	examine	0.597234
poverty	midst	0.197982
letter	book	0.446783
construction	plant	0.502133
identify	adding	0.306849
day	week	0.302847
gray	green	0.212985
varied	shared	0.442941
drank	hay	0.191549
drawn	gray	0.506984
package	hay	0.286869
jail	di	0.234771
convenience	hay	0.212920
taxes	estimated	0.327287
bars	puts	0.251591

6 Clustering

Next V is clustered into 100 groups based on there embeddings via `sklearn.cluster`. Some of the top clusters are shown below:

A cluster commonly used for weather report:

summer spring start winter spent coming sun fall suddenly afternoon saturday instead evening hot sunday started hours opened late cold

A cluster describing international affairs and education:

aid vocational support assistance financial planning services technical defense schools programs training private international research act foreign community medical provide

A cluster talking about production economy:

income estimated net capital annual operating additional average pay property increased farm rates billion price amount stock increase production pressure

And a cluster about military:

staff division army corps activities personnel management facilities services forces association peace research medical committee department force industrial military service