Move to pose with QR code

Yuhao Zhang

April 30, 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 under the guidance with QR codes. The code is for Sunfounder's PiCar-V platform and a webcam attached to it is used.

2 Pose estimation using PnP

With a simple pinhole model the scene view s of a camera is formulated as:

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & c_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix},$$

where (X, Y, Z) is the coordinate of the object point in world reference frame, (u, v) is the coordinates of the projected object on the view. The first factor on the right hand side is the camera matrix which depicts the intrinsic properties of the camera, while the second factor is the rotation-translation matrix that transforms from world reference frame to the camera frame.

The rotation-translation matrix $R - T$ is solvable by first, detecting the QR code using library such as `zbar`. Second, finding the corner points of it and associate with their projections on the view. Once $R - T$ is obtained, one can solve directly for its inverse, as the inverse serves directly for pose estimation of the camera.

Given R which is the rotation matrix, the inverse of it is denoted as $R^{-1} = R^T$. Then the coordinates of the camera in world reference frame are obtained as

$$(x, y, z) = -R^T V.$$

And the distance between the camera and the object:

$$d = \sqrt{(X-x)^2 + (Y-y)^2 + Z-z)^2}.$$

Furthermore, the Euler angles representation can also be acquired easily form R. As for car-like mobile robots, the motion is usually confined to 2-D plane. Consider such a plane is depicted is a right-hand coordinate system as $x-z$ plane and axis y is pointing to the ground. Then only Euler angle θ_y is of interest. Then the problem can be largely simplified. For a landmark $M(X, Y, \gamma)$ in world reference frame, the current pose of the camera is

$$(X - d\cos(\theta_y), Y + d\sin(\theta_y), \pi/2 - \theta_y + \gamma).$$

With this pose estimation, one can combine a control algorithm with feedback from QR codes detected by camera.

A picture showing the performance of such model is plotted as Fig. (1).

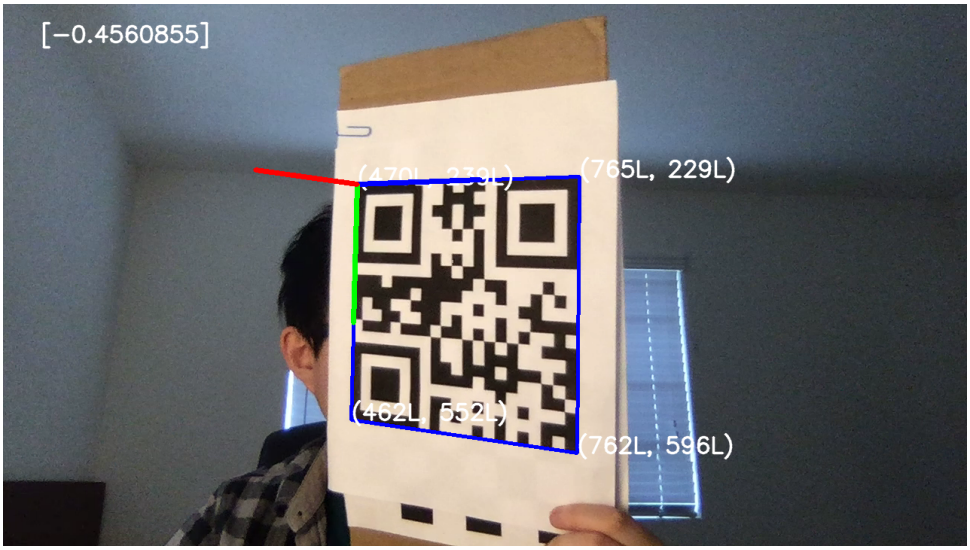

Figure 1: The QR code used as a landmark.

3 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 $v' = -v$.

4 implementation and setup

The control law purposed in Sec.(3) and the pose estimation algorithm purposed in Sec.(2) have been implemented in `QR_code_navi.ipynb`. Given the input `waypoints`, the algorithm does time sampling after every Δt. It utilize the $R-T$ matrix described in Sec.(2) to localize the camera in world frame based on the QR codes scanned. Then the control plan is generated by the algorithm described in Sec.(3). The function `pose_estimator` also accepts precedence of multiple QR codes, in which case the estimated pose is calculated as the mean.

However, this approach suffers from huge deviation as the orientation estimation of camera based on a single QR code is not precise enough. There are typically $\sim 0.2m$ error in terms of position estimation and $\sim 10^\circ$ error in terms of the angle θ_y. Combining with the control algorithm it gives unacceptable errors for localization, as the generated plan tends to be more and more unstable once the localization error begins to manifest.

As an alternate solution, the orientation information of the camera obtained by solving for rotation matrix is discarded, only the position of the camera is used. Furthermore, the time sampling control algorithm is substituted by straight lines plus constant curvature turning that are calibrated at real-time by the current position of the camera.

5 Experiment and conclusion

The algorithm is run on Sunfounder's PiCar-V platform with input `waypoints`.

The error for each points is listed as Tab.(1). The error-bar is given as the 95% confidence interval.

Table 1: Experiment results on the robot

Waypoints	Δ distance/m	Δ radian
(0,0,0)	/	/
(1,0,0)	0.02 ± 0.01	0.03 ± 0.01
(2,2,π)	0.7 ± 0.05	0.24 ± 0.05
(0,0,0)	0.13 ± 0.10	0.32 ± 0.06

The performance has improved much comparing to prj1. With the guidance of QR code the algorithm can achieve almost perfect result on waypoint $(1, 0, 0)$. Yet for (2,2,π) and (0,0,0) it is not as promising.

The algorithm of this experiment only utilizes the coordinates generated by the feedback of QR codes, which means it cannot estimate its current orientation. It is the major reason for the errors. This can also be seen from the error of the orientations.

It might also be possible to use a "track ball" algorithm to guide the car and ignore the PnP problem and frame transformation, as in this task the waypoints are not too complicated. In such a way the accuracy of localization might be improved.