Planning based on Vonoroi graph and visibility graph

Yuhao Zhang

May 25, 2018

1 Introduction

The project is an implementation of EKF-SLAM on Sunfounder's PiCar-V platform and a webcam attached to it is used. With the help of QR codes as landmarks, the webcam is turned into a distance-bearing sensor.

2 Planning based on Vonoroi graph and visibility graph

In this project the EKF-SLAM system will be built from scratch. For a thorough introduction to EKF-SLAM, please check [**?**]. In this project, the landmarks used for SLAM will be QR markers. The SLAM will only consider 2D-plane, therefore the landmarks will be abstracted as points on the plane, although theoretically the QR markers can be used as a 3D object that enables more information and features to be extracted. The detail will be discussed in Sec. **??**.

3 Localization

One major problem for Sunfounder's PiCar-V platform is the lack of a distance-bearing sensor that would make localization with landmarks much easier. However, with the help of known-sized QR tags as 3D object, it is plausible to convert a plain webcam into a distance-bearing sensor.

3.1 Distance and 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

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

3.2 Bearing

The bearing of the object with respect to the camera can be calculated directly using the rotation-translation matrix $R - T$. But in the experiments such calculation turns out to be quite inaccurate and is not sufficient for real application. Therefore in the rest of the paper

the bearing is obtained as

$$(c_x - c_c) \times \frac{d}{f_x},$$

where c_x is the x principle point as shown in the camera matrix, c_c is the center of the QR tag in camera frame and f_x is the $x-$direction focal length of the camera.

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. (**??**).

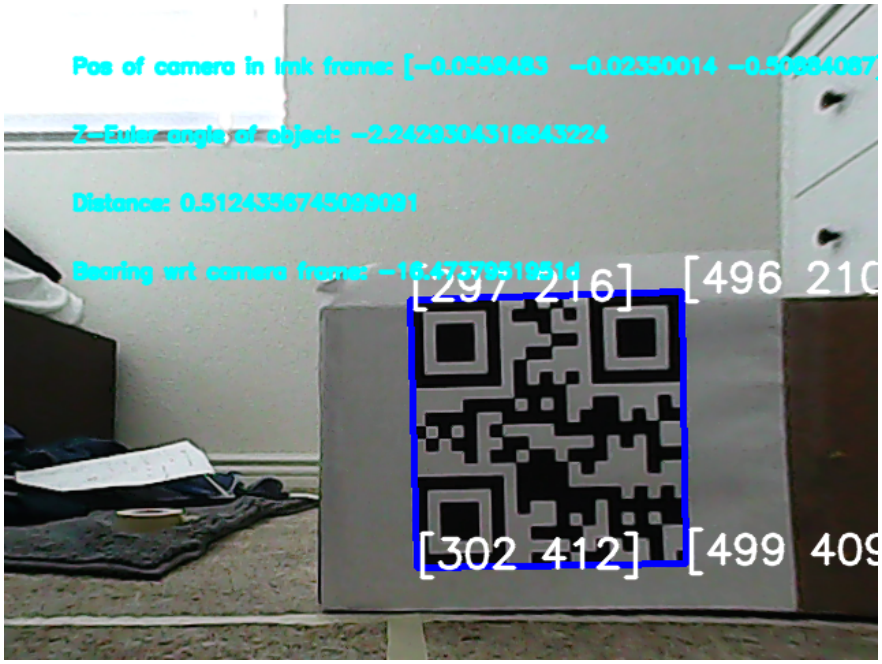

Figure 1: The QR code used as a landmark.

4 Kinematics and control law

The kinematics of the car is orthogonal to the current 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 $v' = -v$.

5 implementation and setup

The EKF-SLAM algorithm purposed in Sec.(**??**) is implemented in library `EKF_SLAM.py` following and being translated from the Matlab code provided in [**?**]. The pose estimation algorithm purposed in Sec.(**??**) has been implemented in library `QRDB.py`. Additionally, a script that calls these libraries is provided as `EKF-SLAM-QR-DB.ipynb`.

After initialization, the algorithm is time sampled every 0.5 seconds and at each step, the camera takes 20 images and runs QR code detection on them. All vision data is precessed for un-distortion. Then the average of the results are taken. The control signal is a constant