AprilTag Localization Technique Document of A3

Wu Chengyu 7086cmd@gmail.com

January 31, 2024

Contents

1 Introduction

The map of the contest is a 2D plane with `apriltags` on it. The `apriltags` are asymmetrical and easy to recognize.

Through the camera, we can fetch all the `apriltags` in the view of the camera. Using `OpenCV`, we can get the position of the `apriltags` in the image. Then we can use the position of the `apriltags` to calculate the position of the camera.

The `ROS` (Robot Operating System) provides apis to get the position of the `apriltags` in the image. However, we also need the camera to recognize other blocks, e.g. the "fish" in the contest. So we deprecated the `ROS` and simply use `Python` with `OpenCV` to get the position of the `apriltags` in order to locate.

2 Methodological Analysis

In general, positioning requires 6 degrees of freedom. That is to say, we need at least 6 points to calculate the accurate position of the camera. However, the 3 degrees of freedom will remain constant during the robot's motion.

For spatial coordinates, we use the Cartesian coordinate system (x, y, z). For the angular coordinates, we use the Euler angle (α, β, γ). Or we can call it $(\text{roll}, \text{pitch}, \text{yaw})$.

The punctuation of `apriltags` is fixed, therefore, as long as we know the position of `apriltags` in the image and its punctuation, it is possible to correspond the two-dimensional image coordinates to the three-dimensional spatial coordinates. Using PnP algorithm, we can get the position of the camera.

2.1 Basic Parameters

We need the camera's internal reference and distortion coefficients to accurately confirm the conversion.

The camera's internal reference is a 3×3 matrix, which is the camera's focal length and the center of the image:

$$\texttt{camera\_matrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

The distortion coefficients are 5 parameters, which are used to correct the distortion of the image:

$$\texttt{distortion\_coefficients} = \begin{pmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{pmatrix}$$

The relevant parameters of the camera are not considered to vary excessively during the process. In other words, we can store these parameters directly at call time.

2.2 6-point Adjustment

Each location of the `apriltags` can define $\frac{1}{6}$ of the camera's position. Therefore, we need at least 6 `apriltags` to calculate the position of the camera.

$$\begin{pmatrix} x & y & z \\ \text{roll} & \text{pitch} & \text{yaw} \end{pmatrix}$$

Through the PnP algorithm provided by `OpenCV`, we can calculate the transform vector and the rotation vector of the camera easily.

After getting `tvec` and `rvec`, we can use the `Rodrigues` function to convert the rotation vector to the rotation matrix:

$$\texttt{R\_mtx} = \texttt{Rodrigues}\,(\texttt{rvec})$$

Then, through some simple calculations, we can get the Euler angle.

2.3 3-point Localization

The `apriltags`'s center point is accurate enough to calculate the position of the camera. That is to say, through at least 3 `apriltags`, we can calculate the position of the camera through the PnP algorithm.

Knowing z, roll and pitch, we should calculate the "bias" matrix.

Define the original matrixes of rotate and transform through these elements:

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}, R_z = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1}$$

Then, we can get the rotate "bias" matrix:

$$R_{\texttt{bias}} = R_x \cdot R_z = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma\cos\alpha & \cos\gamma\cos\alpha & -\sin\alpha \\ \sin\gamma\sin\alpha & \cos\gamma\sin\alpha & \cos\alpha \end{pmatrix} \tag{2}$$

Through `Rodrigues` method, we can get the rotate vector of the "bias" matrix.

Also, the transform vector is easy to get:

$$\texttt{tvec}_{\texttt{bias}} = \begin{pmatrix} 0 \\ 0 \\ -z \end{pmatrix} \tag{3}$$

2.4 1-point Emergency Localization

We can know each corner of the `apriltags` of the map. Therefore, we can calculate the position of the camera through the PnP algorithm.

However, I strongly recommend that we should not use this method. It is not so accurate and it is easy to make mistakes.

If there is less than 3 `apriltags` in the view of the camera, we can calculate corners of the `apriltags` in the image. Then we can calculate the position of the camera through the PnP algorithm.

2.5 Turn Around

The robot can never gonna give you up, never gonna let you down. It can never gonna run around and desert you.

So, don't cry, don't say goodbye, don't tell a lie and hurt you.