Proceedings of the 37th Chinese Control Conference
July 25-27, 2018, Wuhan, China

Pose Estimation for Multicopters Based on Monocular Vision and AprilTag

Guo Zhenglong[1], Fu Qiang[2], Quan Quan[1]

1. School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China
E-mail: guozhenglong@buaa.edu.cn, qq_buaa@buaa.edu.cn

2. School of Instrumentation Science and Opto-electronics Engineering, Beihang University, Beijing 100191, China
E-mail: fq@buaa.edu.cn

Abstract: This paper deals with the problem of pose estimation for multicopters by using a downward-looking monocular camera and AprilTag. First, successive AprilTag images are captured by the downward-looking camera and the relative pose between them is estimated using a vision algorithm directly. Then, a Kalman filter combined with a constant-velocity process model is adopted to improve the accuracy and robustness of pose estimation. Experimental results show that the proposed pose estimation method can satisfy closed-loop control requirements of multicopters.

Key Words: Multicopters; Monocular Camera; AprilTag; Pose Estimation; Closed-loop Control

1 Introduction

Nowadays multicopters have been widely used in many applications. For example, Amazon has designed and tested a future deliver system -- Prime Air to deliver goods using multicopters [1]. During the 2017 Super Bowl halftime show, 300 multicopters called shooting star were employed [2]. Accurate and reliable pose estimation is a fundamental issue for their autonomous control. A commonly-used pose estimation (or navigation) method is Global Positioning System (GPS), but it is not suitable for GPS-denied situations, such as indoor or urban areas. In contrast, vision-based navigation methods do not depend on GPS and could provide high-precise pose within a close range. AprilTag shown in Fig.1 is a kind of visual fiducial system for robotics applications designed by E. Olson [3]. The system is composed of two major components: the tag detector and the coding system. It is designed to be automatically detected and localized even when the tag is at very low resolution, unevenly lit or oddly rotated. If the camera is calibrated and the physical size of the tag is known, it can provide the relative transformation between the tag and the camera. So it can be used to assist camera calibration [4] and is known for their application in augmented reality [5]. Also, with the measuring function of AprilTag, it plays a role in simultaneous localization and mapping (SLAM) [6]. In the literature [7], a robot team is built to explore with the help of AprilTag.

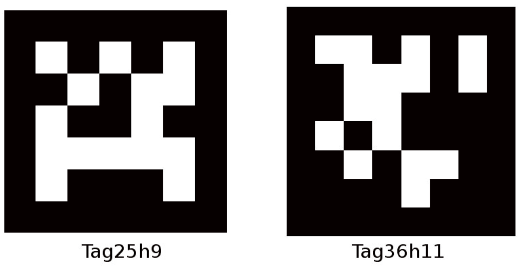

Fig. 1: Examples of different AprilTag families

This work was supported by the Open Project Program of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (No. BUAA-VR-17KF-03)

Therefore, it is natural and convenient to develop a monocular vision method using AprilTag to estimate the pose of multicopters. For a practical system, the positions of AprilTags need to be preset under some specific rules and constraints. In our paper, 400 AprilTags are distributed with the same spacing to compose a map.

In computer vision, determining the relative pose between a 2D-tag and a calibrated camera is often known as a Perspective-n-Point (PnP) problem from n 2D-2D point correspondences [8]. For solving PnP problem, the pose estimation methods for marks (include tags) can be generally classified into three categories : linear methods [8–10], iterative methods [11–13], and recursive methods [14–16]. Linear methods are simple and intuitive but are sensitive to noises. Recursive methods are accurate, efficient and very suitable for image sequence processing with the temporal filtering such as Kalman filters. In the literature [10], the method of pose estimation is a kind of linear methods, and the author developed a supported platform integrated with AprilTags for multicopters indoor autonomous localization. The integrated development environment is the advantage of the platform, but the system architecture has not included the filter to improve the robustness of this system. Therefore, we used the map composed of AprilTags and a downward-looking camera to estimate the pose of multicopters with a Kalman filter based on a linear constant-velocity process model, which is a kind of recursive methods.

The contributions of this paper include that: (i) a vision-based location system is built with the Robotic Operation System (ROS) based on this method using the AprilTag and a monocular camera; (ii) a Kalman filter is adapted to improve the accuracy and robustness of pose estimation. Meanwhile, two experiments are presented, which show that pose estimation using this method can satisfy closed-loop control requirements of multicopters.

This paper is organized as follows. Some preliminaries and problem formulation are introduced in section II. In section III, the main algorithm of pose estimation is presented. The experimental results from the real flight are shown in section IV, followed by the conclusion in section V.

Following notations are adopted in this paper consistent with that in [17]. $\mathbb{R}^n$ denotes Euclidean space of dimension n. $\mathbf{R}_{\mathrm{A}}^{\mathrm{B}}$ and $\mathbf{T}_{\mathrm{A}}^{\mathrm{B}}$ denote rotation matrix and translation vector from coordinate system $\{\mathrm{A}\}$ to coordinate system $\{\mathrm{B}\}$, respectively.

2 Preliminaries and problem formulation

2.1 Coordinate system transformation

Let $\mathbf{P} \in \mathbb{R}^3$, vectors $\mathbf{P}_{\mathrm{A}} = [X_{\mathrm{A}}\ Y_{\mathrm{A}}\ Z_{\mathrm{A}}]^{\mathrm{T}}$, and $\mathbf{P}_{\mathrm{B}} = [X_{\mathrm{B}}\ Y_{\mathrm{B}}\ Z_{\mathrm{B}}]^{\mathrm{T}}$ be the coordinates of $\mathbf{P}$ in two different coordinate systems **A** and **B** respectively. In Euclidean geometry, they satisfy

$$\mathbf{P}_{\mathrm{B}} = \mathbf{R}_{\mathrm{A}}^{\mathrm{B}}\mathbf{P}_{\mathrm{A}} + \mathbf{T}_{\mathrm{A}}^{\mathrm{B}} \tag{1}$$

where $\mathbf{R}_{\mathrm{A}}^{\mathrm{B}} \in \mathbb{R}^{3\times 3}$ is the rotation matrix, and $\mathbf{T}_{\mathrm{A}}^{\mathrm{B}} \in \mathbb{R}^3$ is the translation vector.

Here, several coordinate systems are involved and defined as follows. The camera coordinate system $\{\mathrm{C}\}$ is in the front of the multicopter and not the center of the multicopter, and the camera is pointing down. The body coordinate system $\{\mathrm{B}\}$ is attached with the multicopter, and its origin is the center of the drone with X_{B} axis pointing forward of the drone, Y_{B} axis pointing right of the drone, Z_{B} axis pointing downward of the drone. The coordinate system of the map $\{\mathrm{M}\}$ composed with AprilTags is on the ground surface. The coordinate system of the k-th AprilTag $\{\mathrm{T}_k\}$ is on the surface of the tag, and its origin is the center of the tag with X_{T_k} axis pointing forward, Y_{T_k} axis pointing right, $k = 1, 2, 3...n$.

Fig.2 and Fig.3 show the experimental setup along with the coordinate system. O_{B}-$X_{\mathrm{B}}Y_{\mathrm{B}}Z_{\mathrm{B}}$ denotes the body coordinate system of the multicopter, O_{C}-$X_{\mathrm{C}}Y_{\mathrm{C}}Z_{\mathrm{C}}$ is the camera coordinate system, and O_{M}-$X_{\mathrm{M}}Y_{\mathrm{M}}Z_{\mathrm{M}}$ is the map coordinate system and O_{T_k}-$X_{\mathrm{T}_k}Y_{\mathrm{T}_k}Z_{\mathrm{T}_k}$ is the k-th tag coordinate system. As shown in these figures, O_{B}-$X_{\mathrm{B}}Y_{\mathrm{B}}Z_{\mathrm{B}}$ is parallel to O_{C}-$X_{\mathrm{C}}Y_{\mathrm{C}}Z_{\mathrm{C}}$, and O_{M}-$X_{\mathrm{M}}Y_{\mathrm{M}}Z_{\mathrm{M}}$ is parallel to O_{T_k}-$X_{\mathrm{T}_k}Y_{\mathrm{T}_k}Z_{\mathrm{T}_k}$. Therefore, they satisfy

$$\mathbf{P}_{\mathrm{B}} = \mathbf{P}_{\mathrm{C}} + \mathbf{T}_{\mathrm{C}}^{\mathrm{B}} \tag{2}$$

where $\mathbf{T}_{\mathrm{C}}^{\mathrm{B}} = [l\ 0\ 0]^{\mathrm{T}}$, l denotes the distance between the camera and the center of the multicopter.

$$\mathbf{P}_{\mathrm{T}_k} = \mathbf{P}_{\mathrm{M}} + \mathbf{T}_{\mathrm{M}}^{\mathrm{T}_k} \tag{3}$$

where $\mathbf{T}_{\mathrm{M}}^{\mathrm{T}_k}$ is determined by the rules composing the map, as shown

$$\mathbf{T}_{\mathrm{M}}^{\mathrm{T}_k} = [[\frac{k}{20}]d\ \ (k - 20[\frac{k}{20}])d\ \ 0]^{\mathrm{T}} \tag{4}$$

where d denotes the distance between the AprilTags in the map, assuming that the distance in x-axis is equal to that in y-axis.

We denote the angles ϕ, θ and ψ are the Euler angles. A rotation of ϕ radians about the X-axis is defined as

$$\mathbf{R}_X(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}. \tag{5}$$

Similarly, a rotation of θ radians about the Y-aixs is defined as

$$\mathbf{R}_Y(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}. \tag{6}$$

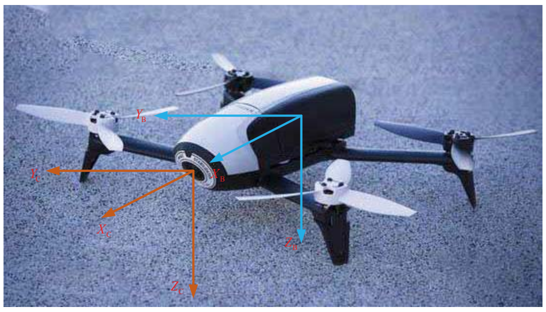

Fig. 2: Camera and body coordinate system

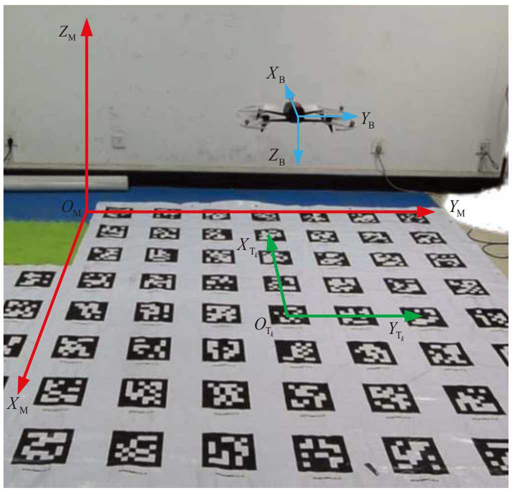

Fig. 3: Some coordinate systems

Finally, a rotation of ψ radians about the Z-axis is defined as

$$\mathbf{R}_Z(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{7}$$

Therefore, the rotation matrix from the map coordinate system to the body coordinate system $\mathbf{R}_{\mathrm{M}}^{\mathrm{B}}$ satisfies

$$\mathbf{R}_{\mathrm{M}}^{\mathrm{B}} = \mathbf{R}_Z(\psi)\mathbf{R}_Y(\theta)\mathbf{R}_X(\phi). \tag{8}$$

The rotation matrix $\mathbf{R}_{\mathrm{T}_k}^{\mathrm{C}}$ can be given as follows

$$\mathbf{R}_{\mathrm{T}_k}^{\mathrm{C}} = \begin{bmatrix} c\varphi c\theta & c\theta s\varphi & -s\theta \\ c\varphi s\phi s\theta - c\phi s\varphi & s\phi s\theta s\varphi + c\phi c\varphi & c\theta s\phi \\ c\varphi c\phi s\phi + s\varphi s\phi & c\phi s\varphi s\theta - c\varphi s\phi & c\phi c\theta \end{bmatrix} \tag{9}$$

where c and s are abbreviations for cos and sin, respectively. According to (2) and (3), the rotation matrix from the k-th tag coordinate system to the camera coordinate system $\mathbf{R}_{\mathrm{T}_k}^{\mathrm{C}}$ satisfies

$$\mathbf{R}_{\mathrm{T}_k}^{\mathrm{C}} = \mathbf{R}_{\mathrm{M}}^{\mathrm{B}}. \tag{10}$$

Since the transform from camera coordinate system to k-th tag coordinate system can be given as follows

$$\mathbf{P}_{\mathrm{T}_k} = \mathbf{R}_{\mathrm{C}}^{\mathrm{T}_k}\mathbf{P}_{\mathrm{C}} + \mathbf{T}_{\mathrm{C}}^{\mathrm{T}_k}. \tag{11}$$

Based on (2), (3), (8), (10) and (11), the transformation from the map coordinate system to the body coordinate system is:

$$\mathbf{P}_{\mathrm{M}} = \mathbf{T}_{\mathrm{T}_k}^{\mathrm{M}} + \mathbf{R}_{\mathrm{C}}^{\mathrm{T}_k}(\mathbf{P}_{\mathrm{C}} + \mathbf{T}_{\mathrm{B}}^{\mathrm{C}}) + \mathbf{T}_{\mathrm{C}}^{\mathrm{T}_k}. \tag{12}$$

Using (12), equations of position parameters can be established and solved.

2.2 Camera pinhole model

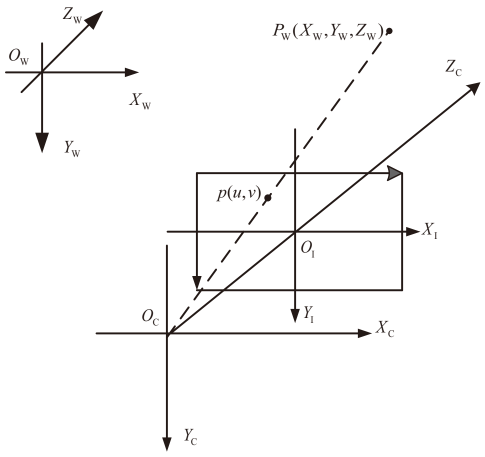

Fig. 4: Camera pinhole model

The camera pinhole model (Fig.4) is used to transform $\mathbf{P}_{\mathrm{C}}$ in $\{\mathrm{C}\}$ and $\mathbf{P}_{\mathrm{T}_k}$ in $\{\mathrm{T}_k\}$ to image coordinate system (O_{I}-$X_{\mathrm{I}}Y_{\mathrm{I}}$) in pixels as equation (13).

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{M}\begin{bmatrix} \mathbf{R}_{\mathrm{T}_k}^{\mathrm{C}} & \mathbf{T}_{\mathrm{T}_k}^{\mathrm{C}} \\ 0^{\mathrm{T}} & 1 \end{bmatrix}\begin{bmatrix} X_{\mathrm{T}_k} \\ Y_{\mathrm{T}_k} \\ Z_{\mathrm{T}_k} \\ 1 \end{bmatrix}, \tag{13}$$

and

$$\mathbf{M} = \begin{bmatrix} \alpha_x & 0 & u_0 & 0 \\ 0 & \alpha_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{14}$$

where s in (13) is the scaling factor; and $\mathbf{R}_{\mathrm{T}_k}^{\mathrm{C}} \in \mathbb{R}^{3\times 3}$ denotes the rotation matrix from the world coordinate system to the camera coordinate system and $\mathbf{T}_{\mathrm{T}_k}^{\mathrm{C}} \in \mathbb{R}^3$ denotes the translation vector. $\mathbf{M}$ is the camera intrinsic matrix with α_x, α_y, u_0, v_0 in $\mathbf{M}$ determined by camera calibration [18].

Taking camera lens distortion into consideration, let (u, v) be the ideal pixel image coordinates, and $(\tilde{u}, \tilde{v})$ the corresponding real observed image coordinates. The ideal points are the projection of the model points according to the pinhole model. Similarly, (x, y) and $(\tilde{x}, \tilde{y})$ are the ideal (distortion-free) and real (distorted) normalized image coordinates. Assuming $\tilde{u} = u_0 + \alpha\tilde{x}$ and $\tilde{v} = v_0 + \beta\tilde{y}$, we have

$$\begin{array}{rcl} \tilde{u} & = & u + (u - u_0)(k_1(x^2 + y^2) + k_2(x^2 + y^2)^2) \\ \tilde{v} & = & v + (v - v_0)(k_1(x^2 + y^2) + k_2(x^2 + y^2)^2) \end{array}, \tag{15}$$

where k_1 and k_2 are the coefficients of the radial distortion. The center of the radial distortion is the same as the principal point.

Note that the parameters $\alpha_x, \alpha_y, u_0, v_0, k_1, k_2$ have no relation to the pose of camera so they are classified as intrinsic parameters. Here, we calibrate the camera with a classical method presented in [18]. $\mathbf{R}_{\mathrm{T}_k}^{\mathrm{C}}$ and $\mathbf{T}_{\mathrm{T}_k}^{\mathrm{C}}$ are determined by the pose of the camera, namely extrinsic parameters, and they are used to estimate the pose of the multicopter.

Here, the projection of the camera is from the 2D tag to a 2D image. Therefore, we use (13) and (15) to compute the relative pose between the tag plane and the image plane. Thus, let $Z_{\mathrm{T}_k} = 0$ in the mode (13), we have the simplified model (17) suitable for our problem.

Here denote

$$\mathbf{R}_{\mathrm{T}_k}^{\mathrm{C}} = [\ \mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{r}_3\ ] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{16}$$

and $\mathbf{T}_{\mathrm{T}_k}^{\mathrm{C}} = [\ t_1 \quad t_2 \quad t_3\ ]^{\mathrm{T}}$, the model (17) can be given as

$$s\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ 1 \end{bmatrix} = \mathbf{M}\begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} X_{\mathrm{T}_k} \\ Y_{\mathrm{T}_k} \\ 1 \end{bmatrix}. \tag{17}$$

The $\mathbf{r}_3$ can be recovered by computing the cross product of the $\mathbf{r}_1$ and $\mathbf{r}_2$, because the columns of a rotation matrix must be orthonormal. Therefore, $\mathbf{R}_{\mathrm{T}_k}^{\mathrm{C}}$ and $\mathbf{T}_{\mathrm{T}_k}^{\mathrm{C}}$ can be computed using the Direct Linear Transform (DLT) algorithm [19].

2.3 Problem formulation

From (17), we get a description of coordinate transformation including pose information, which should be solved using real-time information. By expanding (17) , for every point, we can get

$$\left\{\begin{array}{l} u = \frac{(\alpha_x r_{11}+u_0 r_{31})X_{\mathrm{T}_k}+(\alpha_x r_{12}+u_0 r_{32})Y_{\mathrm{T}_k}+\alpha_x t_1+u_0 t_3}{r_{31}X_{\mathrm{T}_k}+r_{32}Y_{\mathrm{T}_k}+t_3} \\ v = \frac{(\alpha_y r_{21}+v_0 r_{31})X_{\mathrm{T}_k}+(\alpha_y r_{22}+v_0 r_{32})Y_{\mathrm{T}_k}+\alpha_y t_2+v_0 t_3}{r_{31}X_{\mathrm{T}_k}+r_{32}Y_{\mathrm{T}_k}+t_3} \end{array}\right. . \tag{18}$$

The equation (18) can be used to directly to solve our problem. Image coordinates u and v, camera intrinsic parameters α_x, α_y, u_0, v_0 and tag system coordinates X_{T_k} and Y_{T_k} are known. The $\mathbf{R}_{\mathrm{T}_k}^{\mathrm{C}}$ and $\mathbf{T}_{\mathrm{T}_k}^{\mathrm{C}}$ are to be calculated.

In order to solve $\mathbf{R}_{\mathrm{T}_k}^{\mathrm{C}}$ and $\mathbf{T}_{\mathrm{T}_k}^{\mathrm{C}}$, there are more than three feature points in a same image. Thus, the non-linear equations can be solved. From (18), for every feature point in a image, subscript i means the i-th point. Denotes $(\alpha_x r_{11} + u_0 r_{31})X_{\mathrm{T}_k,i} + (\alpha_x r_{12} + u_0 r_{32})Y_{\mathrm{T}_k,i} + \alpha_x t_1 + u_0 t_3$ as num_u , $(\alpha_y r_{21} + v_0 r_{31})X_{\mathrm{T}_k,i} + (\alpha_y r_{22} + v_0 r_{32})Y_{\mathrm{T}_k,i} + \alpha_y t_2 + v_0$ as num_v and $r_{31}X_{\mathrm{T}_k,i} + r_{32}Y_{\mathrm{T}_k,i} + t_3$ as den , We have

$$\boldsymbol{f}(\mathbf{R}_{\mathrm{T}_k,i}^{\mathrm{C}}, \mathbf{T}_{\mathrm{T}_k,i}^{\mathrm{C}}) = \begin{bmatrix} num_u - u \cdot den \\ num_v - v \cdot den \end{bmatrix}. \tag{19}$$

Theoretically, $\boldsymbol{f}$ should be equal to zero vector, $i = 1, 2, 3....$ However, due to the noise and other uncertain factors, we have to solve using an optimization method to get the $\mathbf{R}_{\mathrm{T}_k}^{\mathrm{C}}$ and $\mathbf{T}_{\mathrm{T}_k}^{\mathrm{C}}$. Thus, according to (12), the pose of the multicopter can be calculated.

Next, the proposed method will be introduced in Section III in detail.

3 Main algorithm of pose estimation

3.1 Main algorithm

Here, we define $\boldsymbol{\Theta}_k = [\phi_k \ \theta_k \ \psi_k]^{\mathrm{T}} \in \mathbb{R}^3$ that compose the $\mathbf{R}_{\mathrm{T}_k}^{\mathrm{C}}$, and $\mathbf{T}_k = \mathbf{T}_{\mathrm{T}_k}^{\mathrm{C}} = [x_k \ y_k \ z_k]$.

Since $\mathbf{R}_{\mathrm{C}}^{\mathrm{T}_k} = {\mathbf{R}_{\mathrm{T}_k}^{\mathrm{C}}}^{-1}$, according to (19), we can get the position of the multicopter with the k-th AprilTag using the least square method

$$\mathbf{P}_k = \boldsymbol{g}(\boldsymbol{\Theta}_k, \mathbf{T}_k). \tag{20}$$

Thus, the orientation $\mathbf{O}_k$ of the multicopter can be give as

$$\mathbf{O}_k = \boldsymbol{\Theta}_k. \tag{21}$$

Suppose that there are n tags included in the image captured by the camera of the drone, the IDs of the n tags are $k_1, k_2 ... k_j ..., \ j = 1, 2, 3..., n$. Using (20) and (21), we can get a $(\boldsymbol{\Theta}_{k_j}, \mathbf{T}_{k_j})$ for the k_j tag in the image. According (20) and (21), we have n positions $\mathbf{P}_{k_j}$ and orientations $\mathbf{O}_{k_j}$ of the multicopter, $j = 1, 2, 3..., n$. Finally, we can calculate the average of the n positions and orientations using equations (22) and (23).

$$Pos = \frac{1}{n}\sum_{j=1}^{n} \mathbf{P}_{k_j}, j = 1, 2, ...n \tag{22}$$

$$Att = \frac{1}{n}\sum_{j=1}^{n} \mathbf{O}_{k_j}, j = 1, 2, ...m \tag{23}$$

3.2 Kalman filter

Using the constant-velocity model [15], the process model for estimating the pose of the multicopter can be given as follows

$$\hat{\mathbf{x}}_k = \boldsymbol{\Phi}\hat{\mathbf{x}}_{k-1} + \boldsymbol{\Gamma}\mathbf{w}_{k-1} \tag{24}$$

$$\hat{\mathbf{z}}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k. \tag{25}$$

In equation (24),

$$\mathbf{x}_k = [x_k \ y_k \ z_k \ v_{x,k} \ v_{y,k} \ v_{z,k}]^{\mathrm{T}} \tag{26}$$

is the 6-dimension state vector and

$$\boldsymbol{\Phi} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathrm{dt} \cdot \mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix} \in \mathbb{R}^{6\times6} \tag{27}$$

is the state transition matrix. $\boldsymbol{\Gamma}$ is the noise correlation matrix, and

$$\boldsymbol{\Gamma} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix} \in \mathbb{R}^{6\times6} \tag{28}$$

is the state transition matrix. $\mathbf{w}_{k-1}$ is the Gaussian noise. In equation (25), $\mathbf{z}_k = [x_k \ y_k \ z_k]^{\mathrm{T}}$ is the observation vector.

$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \in \mathbb{R}^{3\times6} \tag{29}$$

is the observation matrix. $\mathbf{v}_{k-1}$ is the Gaussian noise.

Therefore, the Kalman filter consisting of prediction and update parts is given as follows.

$$\begin{gathered} \hat{\mathbf{x}}_{k,k-1} = \boldsymbol{\Phi}\hat{\mathbf{x}}_{k-1,k-1} \\ \mathbf{P}_{k,k-1} = \boldsymbol{\Phi}\mathbf{P}_{k-1,k-1}\boldsymbol{\Phi}^{\mathrm{T}} + \boldsymbol{\Gamma}\mathbf{Q}_{k,k-1}\boldsymbol{\Gamma}^{\mathrm{T}} \\ \mathbf{K}_k = \mathbf{P}_{k,k-1}\mathbf{H}^{\mathrm{T}}(\mathbf{H}\mathbf{P}_{k,k-1}\mathbf{H}^{\mathrm{T}} + \mathbf{R}_k)^{-1} \\ \hat{\mathbf{x}}_{k,k} = \hat{\mathbf{x}}_{k,k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k,k-1}) \\ \mathbf{P}_{k,k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_{k,k-1} \end{gathered} \tag{30}$$

$\mathbf{P}_{k,k-1}$ is a prior covariance of the estimation error, $\mathbf{P}_{k,k}$ is a posterior covariance of the estimation error and $\mathbf{K}_k$ is the Kalman gain matrix at step k. $\mathbf{Q}_{k,k-1}$ is the covariance of the systematic noise. $\mathbf{R}_k$ is the covariance of the observation noise.

After adopting this Kalman filter, we can get a more accurate and reliable result $\hat{\mathbf{x}}_{k,k}$, and we denote $\mathbf{y}_k$ as the final pose of the multicopter. Using

$$\mathbf{y}_k = \mathbf{H}\hat{\mathbf{x}}_{k,k}, \tag{31}$$

the accurate and reliable position and orientation of the multicopter can be estimated.

Therefore, our method of pose estimation for multicopters can be summarized as follows.

Algorithm 1 Procedure of Pose estimation based on AprilTag

Step 1: Set some necessary parameters include the distance d of the tags in X and Y axis in the map, the size of tag and the intrinsic parameters;

Step 2: According to the image captured by the multicopter, measure the camera-relative position and orientation of the tag with the equation (22) and (23);

Step 3: Based on the camera-relative pose, according to (12), calculate the pose of the multicopter;

Step 4: Using the Kalman filter (30) and (31), estimate the accurate and reliable position and orientation of the multicopter.

4 Experiments

4.1 Experiment setup

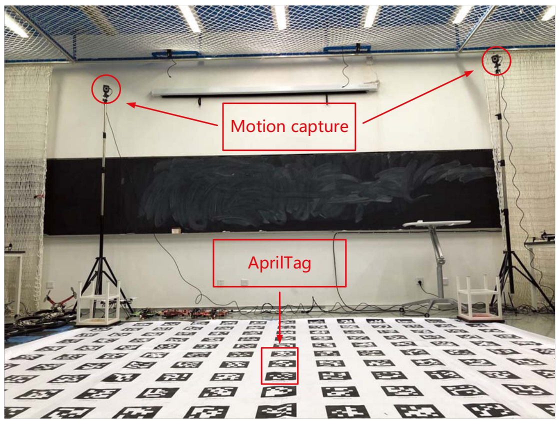

Fig. 5: Experimental scene

As shown in the Fig.5, an experimental system is constructed. This experimental system is constructed with a parrot Bebop2, a map consisted of AprilTags. At the same time, a motion capture system (MoCap) is used to collect accurate data for comparison. The precision of MoCap used in this experiment is high to 1mm. Here we call the proposed method for estimating the pose of multicopters as AprilTags' positioning system (APS) shown in Fig.6 . So the comparison between the pose estimated by MoCap and APS can evaluate the precision of the APS.

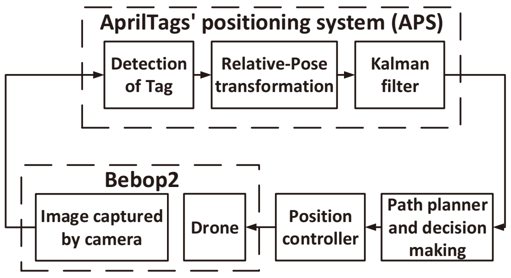
AprilTags' positioning system (APS)
Detection of Tag
Relative-Pose transformation
Kalman filter
Bebop2
Image captured by camera
Drone
Position controller
Path planner and decision making

Fig. 6: The structure of AprilTags' positioning system (APS)

4.2 Detection of AprilTag

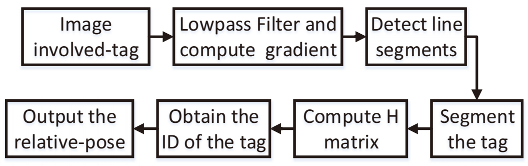

Fig. 7: The flowchart of Tag detection's process

The process of detection of AprilTag is composed of two major components: the tag detector and the code system. The detector is used to find the AprilTag and measure the camera-relative position and orientation of the tag. The code system is to extract the information included in the AprilTag. The process of the detection of the tag is shown in Fig.7.

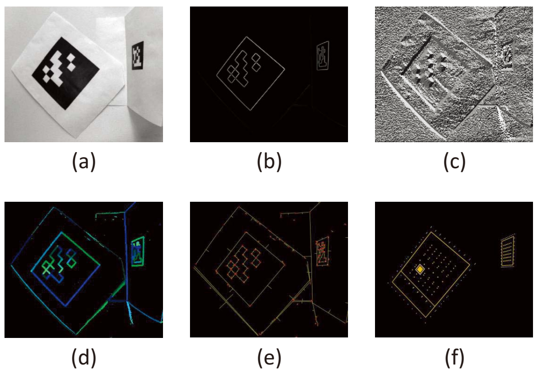

Fig. 8: The process of detecting AprilTag [3]

The Fig.8(a) is the input image. The first step is to detect line segments in the image. As shown in Fig.8(b) and (c), the approach computes the gradient direction and magnitude at every pixel. Using a graph-based method [20], pixels with similar gradient directions and magnitude are clustered into components shown in in Fig.8(d). The next step is to find the square of AprilTag and the line segments. Using weighted least squares, the line segment is fit to the pixels in each component form at last step shown in 8(e). Thus, according to the intersecting lines, the pixels' coordinates can be get and they are at the center of every minor quad in the April-Tag, as shown in Fig.8(f). Then, the homography matrix can be computed using the camera pinhole model (17). The final task is to read the information from the payload field. In this experimental system, the IDs of the AprilTag are the information that we need. All the details are introduced in [3].

4.3 Experiment results

Here, we used the algorithm described in Section III to process the image and estimate the pose of the multicopter based on the images captured by the downward-looking camera.

(1) Comparisons between MoCap and APS

With the pose estimated by the truly pose (from MoCap), we controlled the Bebop2 to fly along four edges of a square. The trajectory of Bebop2 is shown in Fig.9. The red line denotes the value of pose from MoCap, and the blue line denotes the value of pose estimated by APS. The comparisons in X-axis and Y-axis are shown in the Fig.10 and Fig.11. From these figures, we can conclude that the pose estimated by APS is correct and the precision of the value is about 0.02m in the x-axis and y-axis.

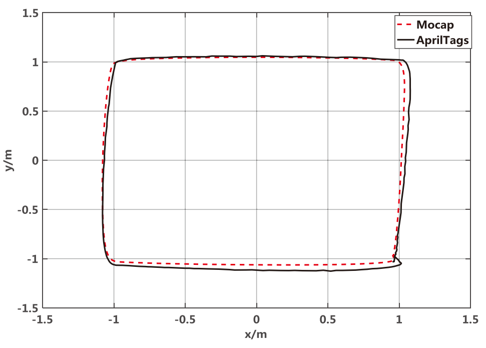

Fig. 9: The trajectory estimated by APS and MoCap

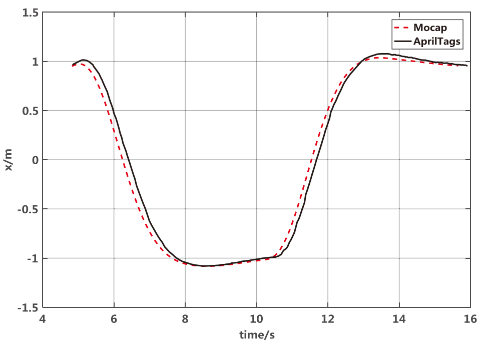

Fig. 10: The position estimated by APS and MoCap in x-axis

(2) Closed-loop control

In this experiment, we conduct a closed-loop control experiment for Bebop2 using APS. The video about this experiment can be found at `http://rfly.buaa.edu.cn`. The task of this flight is to make a traversal from A-B-C-D-A under the certain sequence. The coordinates of these four points are A(0.55,-0.55), B(-0.55,-0.55), C(-0.55,0.55) and D(0.55,0.55). The experimental result in the Fig.12 shows the Bebop2 have arrived every point under the closed-loop

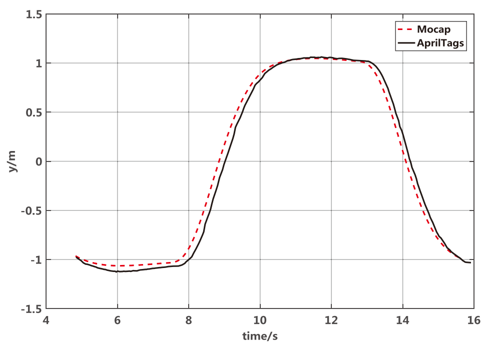

Fig. 11: The position estimated by APS and MoCap in y-axis

control. Because the task of this control is just to make the traversal four points and doesn't include the trajectory control. The result demonstrated that the APS Based on the monocular camera and AprilTag could use to make the closed-loop control for multicopters.

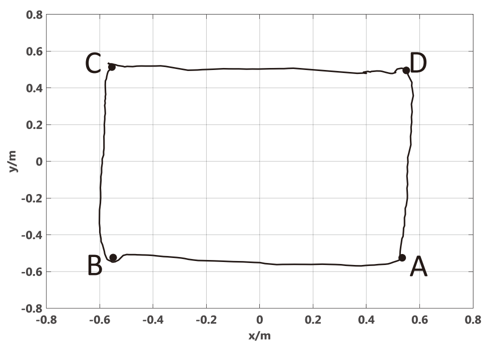

Fig. 12: The trajectory of closed-loop control

Although the algorithm and experiments are accurate, position and attitude information contain some errors. These errors may ascribe these following reasons:

1) The distortion of the camera of Bebop2 cannot be eliminated totally through calibration, so the image system coordinates will occur small errors.

2) When the Bebop2 is flying, a high-frequency vibration will be arised. This will affect the stability of camera and the estimation of the attitude of Bebop2.

3) Because the image captured by the camera is needed to be sent to ground station with WiFi connector, the time delay will generate a little hysteresis on pose estimation.

5 Conclusion

A low-cost localization system based on AprilTag and monocular vision is proposed in this paper. According to the result of these experiments, the system can provide accurate and robust pose estimation for multicopters using the Kalman filters. The experiments have demonstrated that the method for pose estimation is concise and can satisfy closed-loop control requirements of multicopters. Therefore, it is easy to build the localization system for some robots equipped with a camera. Therefore, it is also suitable for researching multiple drones flight if it uses this method of pose estimation.

References

[1] Amazon Prime Air, 2016. [Online]. Available: http://www.amazon.com/ b?node=8037720011

[2] Intel Drones, 2017. [online]. Available: https://newsroom.intel.com/news-releases/intel-drones-light-lady-gaga-performance-pepsi-zero-sugar-super-bowl-li-halftime/

[3] E. Olson, AprilTag: A robust and flexible visual fiducial system[C]//Robotics and Automation (ICRA), 2011 IEEE International Conference on. IEEE, 2011: 3400-3407.

[4] A. Richardson, J. Strom, E. Olson, AprilCal: Assisted and repeatable camera calibration[C]//Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on. IEEE, 2013: 1814-1821.

[5] S. Dong, A. H. Behzadan, F. Chen, Collaborative visualization of engineering processes using tabletop augmented reality[J]. Advances in Engineering Software, 2013, 55: 45-55.

[6] Y. Li, S. Li, Y. Ge, A biologically inspired solution to simultaneous localization and consistent mapping in dynamic environments[J]. Neurocomputing, 2013, 104: 170-179.

[7] E. Olson, J. Strom, R. Goeddel, Exploration and mapping with autonomous robot teams[J]. Communications of the ACM, 2013, 56(3): 62-70.

[8] S. Li, C. Xu, M. Xie, A robust $O(n)$ solution to the perspective-n-point problem[J]. IEEE transactions on pattern analysis and machine intelligence, 2012, 34(7): 1444-1450.

[9] A. Ansar, K. Daniilidis, Linear pose estimation from points or lines[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003, 25(5): 578-589.

[10] S. Wang, T. Hu, ROS-Gazebo Supported Platform for Tag-in-Loop Indoor Localization of Quadrocopter[C]//International Conference on Intelligent Autonomous Systems. Springer, Cham, 2016: 185-197.

[11] M. Faessler, E. Mueggler, K. Schwabe, A monocular pose estimation system based on infrared leds[C]//Robotics and Automation (ICRA), 2014 IEEE International Conference on. IEEE, 2014: 907-913.

[12] C. P. Lu, G. D. Hager, E. Mjolsness, Fast and globally convergent pose estimation from video images[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(6): 610-622.

[13] A. Assa, F. Janabi-Sharifi, Virtual visual servoing for multicamera pose estimation[J]. IEEE/ASME Transactions on Mechatronics, 2015, 20(2): 789-798.

[14] W. J. Wilson, C. C. W. Hulls, G. S. Bell, Relative end-effector control using cartesian position based visual servoing[J]. IEEE Transactions on Robotics and Automation, 1996, 12(5): 684-696.

[15] F. Janabi-Sharifi, M. Marey, A kalman-filter-based method for pose estimation in visual servoing[J]. IEEE Transactions on Robotics, 2010, 26(5): 939-947.

[16] A. Assa, F. Janabi-Sharifi, A robust vision-based sensor fusion approach for real-time pose estimation[J]. IEEE transactions on cybernetics, 2014, 44(2): 217-227.

[17] Q. Quan, Introduction to multicopter design and control[M]. Springer, 2017.

[18] Z. Zhang, A flexible new technique for camera calibration[J]. IEEE Transactions on pattern analysis and machine intelligence, 2000, 22(11): 1330-1334.

[19] R. Hartley, A. Zisserman, Multiple view geometry in computer vision[M]. Cambridge university press, 2003.

[20] P. F. Felzenszwalb, D. P. Huttenlocher, Efficient graph-based image segmentation[J]. International journal of computer vision, 2004, 59(2): 167-181.