Object Recognition to Refine Drone Positioning

Dmitriy Kritskiy
Information technology
National Aerospace University, KhAI
Kharkiv, Ukraine
d.krickiy@khai.edu

Olha Pohudina
Information technology
National Aerospace University, KhAI
Kharkiv, Ukraine
ok.pogudina@gmail.com

2020 IEEE 15th International Conference on Computer Sciences and Information Technologies (CSIT) | 978-1-7281-7443-3/20/$31.00 ©2020 IEEE | DOI: 10.1109/CSIT49958.2020.9321945

***Abstract*—This article is about using object recognition to more accurately position drones. A description is given of using an autopilot with a Kalman filter to reduce the error in positioning the drone. The use of a quaternion for positioning a drone is described. The implementation of the marker recognition module based on the calculation of moments using the OpenCV library and the Python programming language is described, which allows to find the offset relative to the marker, as well as allowing to calculate the positioning error over time. This allows you to synchronize the data received from the camera and autopilot to reduce the error in positioning the drone.**

Keywords—marker recognition; Kalman filter; visual navigation; HSV; FPV camera

I. INTRODUCTION

One of the main areas of research on the creation of small unmanned aerial vehicles (SUAV) - drones, today, is the task of stabilizing the movement and the exact execution of maneuvers, maintaining the position of the drone in space. Recently, significant progress has been made in multicopter flight dynamics control systems, which allowed the use of aggressive maneuvers, drone tennis, and the solution of the group flight problem corrected by the operator. At the same time, these systems require an external motion monitoring system and cannot be used in autonomous drone flight tasks. At the necessary moment, the operator can make adjustments to the position, movement and cancellation of the mission. Effective autonomous flight control is still an open scientific problem, especially where it is impossible to use external positioning systems - such as GPS, GLONASS. The main task that makes up this problem is the complexity of high-quality positioning using only built-in sensors.

Currently, drone positioning is mainly due to the use of the Kalman filter, data received from on-board sensors, use of GPS, etc. [1]. But taking into account the necessity of walking the drone over a certain object or the location of the drone in a certain place relative to the object, this task can be solved using object recognition using data from the camera on board the drone.

For a more convenient use of information from various sensors installed on board and the analysis of the video stream from the camera, the OpenCV library and the Python programming language are most often used [2].

The architecture of the software application in this case can be described as follows. There is an interaction of the Python programming language, the OpenCV library, designed to work with the image received from the camera and something that is computing power, on which the software application is respectively deployed.

II. SUAV NAVIGATION SYSTEMS

The model of the inertial navigation system includes a Kalman filter, which is designed to combine data from several sensors and allows using comparisons to obtain estimates of the position, direction and general dynamics of motion, thereby reducing the error.

The Librepilot project uses an improved orientation estimation algorithm based on the Advanced Kalman Filter (EKF). The need to use this filter is due to the use of a triaxial gyroscope based on built-in angular velocity sensors and triaxial accelerometers, which leads to zero bias. In this regard, the system is nonlinear [3]. For forecasting and correction of estimates, the expansion of nonlinear functions of Taylor series is used with restriction to linear terms [4].

The extended Kalman filter is a multicopter state vector (x_k), which is considered as a continuous dynamic system with discrete measurements, at a time instant it is defined as follows [1,5]:

$$x_k = \begin{bmatrix} p_k \\ v_k \\ q_k \\ b_k \end{bmatrix} = \begin{bmatrix} v_{k-1} \\ R_{eb}(q_{k-1})\alpha_{k-1} \\ \frac{1}{2}\Omega(q_{k-1})\omega_{k-1} \\ w_{b,k-1} \end{bmatrix} \quad (1)$$

where: p_k – three-dimensional (3-D) position of the SUAV at the k moment of time; v_k,v_{k-1} – speed in a fixed coordinate system at current k and previous $k-1$ moments of SUAV state determination; q_k, q_{k-1} quaternions describing the velocities in the Earth coordinate system relative to the SUAV coordinate system; b_k – gyro deflection; $R_{eb}(q)$ and $\Omega(q)$ – rotation matrices; α – linear acceleration; ω – angular velocity in a motionless body; $w_{b,k-1}$ – gyro deflection noise at $k-1$ moment of time.

In formula (1) the gyro deflection b_k is modeled by $w_{b,k-1}$ noise. The system input u consists of angular velocity measurements ω_m and linear acceleration α_m:

$$u_k = \begin{bmatrix} \omega_{m,k} \\ \alpha_{m,k} \end{bmatrix} = \begin{bmatrix} \omega_{m,k} - w_{\omega,k} + b_k \\ \alpha_{m,k} - w_{\alpha,k} - R_{eb}^T(q_k)[0..0g]^T \end{bmatrix} \quad (2)$$

where: w_ω and w_α – represent noise and g – gravitational acceleration. The substitution of (2) in (1) gives the following nonlinear model:

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1} = \begin{bmatrix} v_{k-1} \\ R_{eb}(q_{k-1})(\alpha_{m,k-1} + w_{\alpha,k-1}) + [0..0g]^T \\ \frac{1}{2}\Omega(q_{k-1})(\omega_{m,k-1} + w_{\omega,k-1} - b_{k-1}) \\ w_{b,k-1} \end{bmatrix} \quad (3)$$

where: $w_k = [w_{\omega,k}, w_{\alpha,k}, w_{b,k}]^T$ is the noise of the process.

The nonlinear measurement model has the following form:

$$z_k = h(x_k) + v_k = \begin{bmatrix} p \\ v \\ m_b \\ h_b \end{bmatrix} = \begin{bmatrix} p \\ v \\ R_{eb}^T(q) m_e \\ -P_z \end{bmatrix} \quad (4)$$

where: m_b – measurement of the Earth magnetic field; m_e – in the frame of hull, h_b – height, measured by barometer, P_z and v_k – measurement noise.

Simulink was used to simulate the operation of an algorithm based on EKF; it allows using structural diagrams in the form of oriented graphs for constructing dynamic models, including discrete, continuous, hybrid, nonlinear, and discontinuous systems. Existing model was used for modeling fig. 1 [6].

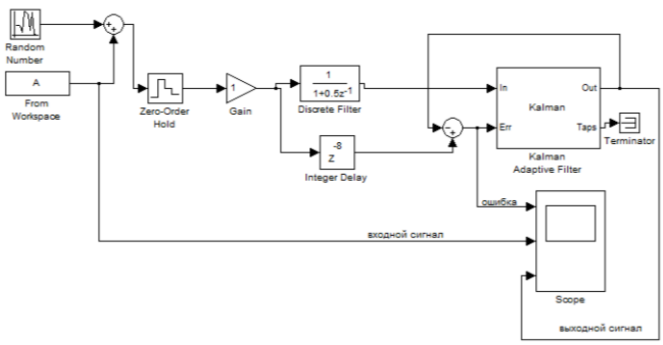

Fig. 1. Block diagram of Kalman filter modeling

For the simulation, accelerated curvilinear motion (angular velocity of 1 m/s^2 and 5 deg/s for each axis) in the time interval of 300 seconds was assumed. The simulation parameters are given in Table. 1. To obtain statistical characteristics, 100 implementations were performed.

The averaging method was used to find the root-mean-square (RMS) error of the estimates for the coordinates, speed, zero bias of the gyroscope and accelerometer. The simulation results, which were obtained by using EKF algorithm are presented in Table. 2.

TABLE I. THE SIMULATION PARAMETERS

Parameter/Sensor	Gyroscope	Accelerometer
Scale factor	0.35%	1.1%
Bias	0.062 deg/s	-0.032 m/s2
Non-orthogonality of axes	0.016 deg	-0.018 deg
RMS of noise	0.018 deg/s	0.055 m/s2

As the result of the simulation, it is concluded that for a certain matrix of initial conditions, the value of the forgetting factor in the bias filter increases monotonically, eventually leading to a divergence between obtained estimates. Accordingly, in order to avoid such errors, it is necessary to limit the forgetting factor in the bias filter.

TABLE II. THE SIMULATION RESULTS

RMS of the coordinate estimation	1.93
RMS of the speed estimation	0.0058
RMS of the gyroscope zero bias estimation, deg/s	0.0106
RMS of the accelerometer zero bias estimation, m/s2	0.031

In addition, for automatic flight it is necessary to use visual data to correct the flight task in case of interference. The shared usage of inertial and visual measurements can be divided into three types:

Application of one type sensor results measurement, as a correction to the measurements results of another type sensor;

Usage of only some inertial measurements together with the results of visual observations;

Combining the results of inertial and visual measurements to improve the definition of position.

III. THE SOFTWARE FOR SUAV NAVIGATION

Let's take a look at Ground Control Interface (GCI). GCI is a wireless conjugate human-robotic interface, designed to monitor and control the SUAV. Usually GCI is aimed at reducing the workload of the operator. There are a lot of open source GCI for SUAV. In case of Librepilot CC3D there is an official cross platform GCI written in C ++ using Qt Framework3 - LibrePilot GCI2. For the user interface, it uses its own specially designed communication protocol UAVTalk. In addition, there is an Android application version of LibrePilot2Go.

Among the limitations of these GCI applications are the following:

- Can control only one SUAV and therefore is not suitable for controlling several SUAVs (group flight);
- Does not support automatic takeoff / landing;
- Data logging start and stop possibility is absent;
- There is no stop and restart of autonomous high-level tasks;
- There are no reactions to emergency situations or interferences, which occur during the task execution.

Thus, to complement the existing GCI, you must use external libraries that are focused on working with the video stream. The OpenCV library was widely distributed. The methods used to determine the key features of the image (SURF – Speeded-Up Robust Features), different classifiers with Haar attributes are used among other things to position SUAV [7,8].

For example, in the SUAV navigation block, a point with SURF on the image received from the camera is determined and, as a result, its allowed area of deviation from the specified location is determined [9]. The Detect() method of the SURF Feature Detector class is used to detect key points on frames from the image of the video camera. And the

similarity of the characteristics is compared with the use of the match() method of the BFMatcher class. The image homography is calculated using the function findHomography(). Location of SUAV and its rotation is calculated using the perspective Transform() function.

A system for visual navigation of the SUAV was developed for determining the distance to the marker, entering this information into the storage and further movement behind the recognized marker, at a safe distance. The SUAV orientation is performed by using the FLANN algorithm, since it is implemented in the OpenCV library [10]. While the program is running, the control panel interacts with the multicopter and receives the data about the current position of the marker. The user interface for controlling the multicopter contains the following windows: the main window which contains data about the on-board equipment performance, GPS coordinates of the SUAV, telemetry data, the presence of streaming video, as well as the settings of the SUAV; the window with the marker image in thermal format; the window with contours for determining the size of the marker and the distance between SUAV and the marker.

The developed system provides for the use of altitude data obtained from the aircraft's avionics. Initial testing was carried out with a webcam connected to a computer and a camera installed on a laptop. Therefore, the program also provides manual input of the distance to the object under study.

After entering the distance, a second settings window appears, designed to enter the ranges of the color component in the HSV format, which allows you to adjust the color of the marker under study, namely, to indicate in which color range the marker color is located.

The color encoding of the HSV model is used: H - color tone, S - saturation, V - brightness. Therefore, to highlight by color, it is enough to indicate the H range, and the saturation and brightness will vary greatly (due to different illumination, oblique angles of the object, etc.). The following settings were used in the experiment, if H is 80 (green), the lower limit is set to H = 80-10 [70, 50, 50], and the upper one is H = 80 + 10 [90,255,255].

After the sliders are set to the desired positions, the third and fourth windows appear, designed to display a monochrome image (where white color is shown on a black background, the number and location of a given color) and to display the image received from the camera on which the marker is located and the coordinates in which is the center of the marker. It should be noted that the coordinates are displayed relative to the upper left corner, as well as measurements are carried out in centimeters (there is the possibility of changing units to meters). Also, for convenience, the number of frames processed per second is shown in the upper left corner.

The standard technique in OpenCV to solve the problem consists of two stages.

At the first stage, we apply a color filter and turn each frame into a black and white picture. After applying the filter, an object of a given color turns into a white spot, and everything else is filled with black.

In the next step, we use the moment calculation algorithm. The image moment is the total characteristic of a spot, which is the sum of all the points (pixels) of this spot. At the same time, there are many types of moments that characterize the different properties of the image.

For example, the moment of zero order m00 is the number of all points that make up the spot. The first-order moment m10 is the sum of the X coordinates of the points, and m01 is the sum of the Y coordinates. There are also moments m11, m20, m02, m22, etc.

The formula for calculating the moments is very simple, and we can count the pixels manually. However, the standard OpenCV functions are written in lower-level languages than Python and are faster. We use the standard function for calculating frame moments: moments (frame, binary).

The argument frame is our pre-processed picture. The binary argument determines how the algorithm will calculate the weight of each point. Let me remind you that the previous inRange method gave us a black and white picture in which pixels can be black, white, or maybe gray. So, if the binary argument is 1, then the weight of all points with a color other than zero will be equal to one.

The moments function will return an array of moments up to the third order. To get the X and Y coordinates of the desired spot, we should divide the obtained moments m10 and m01 by the zero moment m00. Thus, we will find the average coordinates X and Y of all points, and this is the center of the spot.

In our case, the algorithm is complicated by the additional introduction of the distance at which the object is located, necessary to calculate the coordinates at which the object is located, as well as by displaying the coordinates of the object at which it is located.

Fig. 2 shows the interaction of the program modules with the OpenCV library, where: Main is the main window of the program; CannyI is the window with contours of all objects that are within the camera's view and finally, CannyY is the window with the marker image in thermal format.

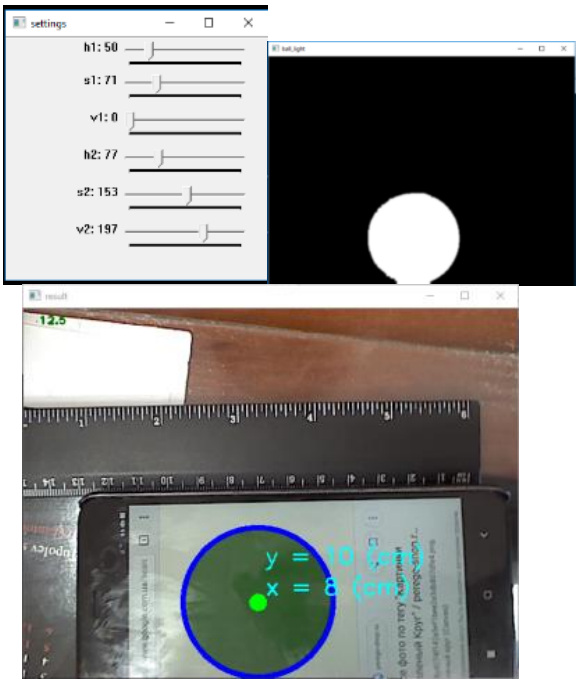

Fig. 2. Interactions of program subsystems

After testing the program, it was determined that there is an error of 0.01-0.03 m in recognizing the distance between the SUAV and the marker. This error is not significant, therefore, by determining the distance to the marker, the SUAV coordinates can be considered as known. The limitation is that the SUAV will move at a given altitude behind the object (in our case by a robot on wheels), with a fixed marker. A red ball with the diameter of 0.07 m was chosen for this experiment.

It is necessary to introduce a new position that appears in our results: the local position. This is the position that is integrated by the flight controller which it determines based on the position the onboard computer provides and his internal sensors. His movements will only be based on that estimated position. This position should be consistent with the position we give it, since the first is a filtered version of the later.

All reader threads are launched at the start of the program. The first thread reads the total station data. It opens a socket through USB and waits for data to arrive. The second thread reads the information from the camera, applies the transformations to the positions, and updates its own message.

The thread loops at a constant rate (20 Hz). Each cycle, it picks the best currently available source for both positions, and uses the camera's orientation. This pose is then sent to the controller unit (Local Position Estimator. Extended Kalman filter for 3D position estimation).

IV. CONCLUSION

With this article, we presented our positioning systems: the total station and the cameras. Those technologies were integrated together in the positioning system through a strategy that chooses the most trustworthy available source for position and orientation in order to send a unique information to the drone that is as accurate as possible.

In this article, the structure of SUAV navigational systems that perform the tasks of an autonomous flight with a duration of up to 0.5 hours with a weight of up to 7 kg is considered. The task of positioning is solved both with the help of built-in (gyroscope, accelerometer, barometer, magnetometer) and with plug-ins (GPS receiver). The task of navigation and interaction with the obstacles encountered is solved based on the video stream processing received from the FPV camera installed.

The inertial navigation systems considered in this article have a measurement error, which is compensated by algorithm using the Kalman filter. The simulation results show an error in determining the multicopter coordinates, which increases monotonically with time.

The structure of the connected systems to the multi-copter autopilot is proposed, which allows to implement autonomous flight control by taking into account the data of inertial navigation system, satellite communication and optical system. The navigation task is assigned to the following subsystems: MPU-6000, MS5611 barometer, Honeywell HMC5883L magnetometer, OpenPilot GPS v8 receiver. The orientation task (altitude control) will be solved using a video camera with signal processing by the Orange Pi PC module.

The approach proposed in the article is based on two components: using the Kalman filter to reduce data errors received from sensors on board the drone, as well as to use the offset calculation relative to the marker to correct the drone position. The development of this approach will increase the accuracy of the drone passing over a specific place.

REFERENCES

[1] H. Lim, J. Park, D. Lee, H. J. Kim, "Build your own quadrotor: Open-source projects on unmanned aerial vehicles," IEEE Robotics & Automation Magazine, vol. 19, pp. 33-45, 2012.

[2] D. Kritskiy, A. Karatanov, S. Koba, E. Druzhinin, "Increasing the reliability of drones due to the use of quaternions in motion," IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT), pp. 348-352, 2018.

[3] A. S. Konakov, V. V. Shavrin, A. A. Savin, V. I. Tislenko, "Issledovanie statisticheskih svojstv ocenok koordinat v besplatformennoj inercial'noj navigacionnoj sisteme s ispol'zovaniem kvaternionnogo metoda preobrazovanija bazisov," Doklady Tom. gos. un-ta sistem upravlenija i radiojelektroniki, vol. 2, no. 24, pp. 49-53, December 2011.

[4] S. J. Julier, J. K. Ulhmann, "A new method for nonlinear transformation of means and covariance's in filters and estimators," IEEE Transactions on Automatic Control, vol. 45, pp. 472–478, March 2000.

[5] H. Chen, X.-M. Wang, Y. Li, "A survey of autonomous control for UAV", International Conference on Artificial Intelligence and Computational Intelligence, vol. 2, pp. 267-271, November 2009.

[6] P.M. Shonazarov, F.T. Kholov, O.O. Evsyutin, & U.A. "Tursunbadalov, Application of Kalman filter in the task of technical diagnostics of internal combustion engines," Bulletin of the South Ural State University. Series: Computer Technologies, Management, Electronics,vol. 19 (1), pp. 152-159, 2019.

[7] A. Mojsilovic, J. Kovacevic, J. Hu, R. J. Safranek, S. K. Ganapathy, "Matching and retrieval based on the vocabulary and grammar of color patterns," IEEE Transactions on image processing, vol. 9, no. 1, pp. 38-54, 2000.

[8] H. Fleyeh, "Color detection and segmentation for road and traffic signs," IEEE Conference on Cybernetics and Intelligent Systems, vol. 2, pp. 809-814, 2004.

[9] N. I. R. Pacheco, D. D. C. R. Resende, P. A. A. Magalhães, "Stability Control of an Autonomous Quadcopter through PID Control Law," International Journal of Engineering Research and Application, vol. 5, pp.7-10, May 2015.

[10] D. A. Kuplyakov, E. V. Shalnov, V. S. Konushin, and A. S. Konushin, "A Distributed Tracking Algorithm for Counting People in Video," Programming and Computer Software, vol. 45, no. 4, pp. 163-170, 2019.