Sensor-Fusion-Based Trajectory Reconstruction for Quadrotor Drones

Jielei Zhang(✉), Jie Feng, and Bingfeng Zhou

Institute of Computer Science and Technology, Peking University, Beijing, China
{zjl1992,feng_jie,cczbf}@pku.edu.cn

Abstract. In this paper, we propose a novel sensor-fusion-based method to eliminate errors of MEMS IMUs, and reconstruct trajectory of quadrotor drones. MEMS IMUs are widely equipped in quadrotor drones and other mobile devices. Unfortunately, they carry a lot of inherent errors, which cause poor results in trajectory reconstruction. To solve this problem, an error model for accelormeter signals in MEMS IMUs is established. In this model, the error is composed of a bias component and a noise component. First, a low-pass filter with downsampling is applied to reduce the noise component. Then, the bias component is detected and eliminated dynamically with the assistance of other sensors. Finally, the trajectory of the drone is reconstructed through integration of the calibrated accelormeter data. We apply our trajectory reconstruction method on Parrot AR.Drone 2.0 which employs a low-cost MEMS IMU. The experimental results prove its effectiveness. This method can theoretically be applied to any other mobile devices which are equipped with MEMS IMUs.

Keywords: MEMS IMUs · Sensor fusion · Inertial navigation · Trajectory reconstruction

1 Introduction

As one kind of mobile devices, the quadrotor drone has the characteristics of wide application, small size, flexible movement, and so on. A large amount of methods have been proposed for the application of comsumer-level drones these years. Trajectory reconstruction is one of the most important basis for drone applications.

The trajectory reconstruction has a wide application in computer graphics. The self-location information can provide important camera parameters, which could be applied in 3-D reconstruction [1] or image-based rendering. It also can be applied in self-localization and map building [2]. Additionally, the drone cinematography could utilize reconstructed trajectory as virtual rail [3].

Methods based on IMU (inertial measurement unit) is one of options for trajectory reconstruction. The position can be calculated through the second-order integration of accelerometer signals in IMU Data [4].

D. Bechmann et al. (Eds.): VISIGRAPP 2018, CCIS 997, pp. 3–24, 2019.
https://doi.org/10.1007/978-3-030-26756-8_1

However, due to the character of MEMS, significant errors occur in measured IMU data, especially low-cost MEMS IMU. Those errors usually take the form of *noise* and *bias* [5].

A number of methods have been proposed to reduce errors in MEMS signals [6–8]. For instance, Kalman Filter is a common estimation approach. Usually, through Kalman Filter, IMU data is combined with computer vision, GPS or other sensors, which can also reconstruct trajectory directly or indirectly. However, the reconstruction results through Kalman Filter depends on the error-staute estimation, which is estimated in advance. Such methods may fail, if the error statue is estimated incorrectly.

Take those limitations into consideration, we focus on the trajectory reconstruction based on IMU. An effective method is proposed to reduce errors in MEMS sensors in this paper.

We establish an error model, which consists of the types of MEMS errors, i.e., the noise and the bias. Different from traditional methods that eliminate bias just once at the static beginning, we process the two components separately and dynamically without highly depending on the priori error-state estimation. For the noise component, errors are treated as high-frequency signals, which can be reduced by a low-pass filter with downsampling. For the bias componet, errors occur in a form of *data drifting*. Thus, we adopt a sensor-fusion method to detect and eliminate the bias. We first detect the event timestamps with the help of multiple auxiliary sensors. The accelerometer data is then segmented into sections according to those timestamps, and the bias is corrected in each section. After that, the calibrated data is obtained.

The pipeline of our method is as follows: First, IMU sensor data is collected and preprocessed, including coordinate transformation between body frame and inertial frame. Then, errors, both noise and bias, are eliminated according to the established error model. Finally, the trajectory is reconstructed through the second-order integration of calibrated accelerometer data. The effectiveness of our method is proved by experimental results.

The main contributions of this paper include:

1. A novel method aiming to estimate errors in MEMS is proposed in this paper. Errors in different types are eliminated seperately and dynamically. This method relies less on priori error estimation.
2. The sensor-fusion-based bias elimination algorithm in our paper is highly adaptive. The type of combined sensors is not limited to what we applied in this paper. In fact, it can be extended to any other sensors.
3. Our method works effectively even on low-cost MEMS IMU which produces more instable errors, while most previous work relies on MEMS IMU with higher precision.
4. In theory, our method can be applied not only to quadrotor drones but also to other mobile devices, as long as they are equipped with MEMS IMU.

2 Related Work

2.1 Trajectory Reconstruction Method Based on Sensors

Plenty of sensors, equipped on mobile devices, can be used in trajectory reconstruction. There are lots of trajectory reconstruction methods based on sensors, such as follows:

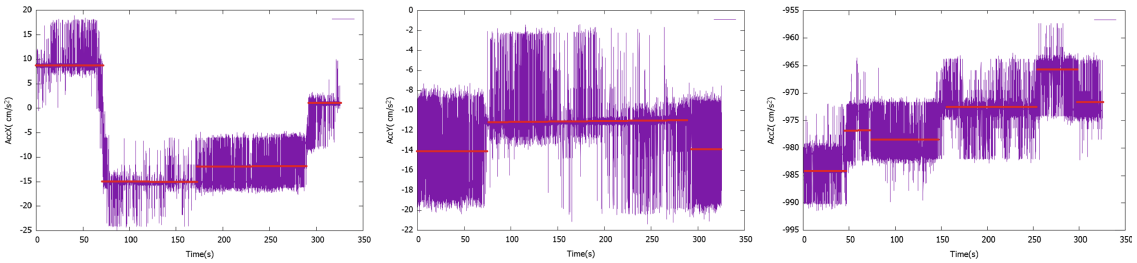

Fig. 1. Raw accelerometer signals of 3 axes (X, Y and Z) collected from a static device. The significant noise and data drifting in the signal will lead to wrong results in trajectory reconstruction [30]. (Color figure online)

GPS-Based Localization. The principle of localization method based on GPS is that, a GPS receiver monitors several satellites and solves equations to determine where the receiver is in real time. Due to the low accuracy and limitations of GPS, many methods are raised to improve accuracy, such as D-GPS [9], aided navigation [10] and so on. However, GPS is not available around large obstacles such as tall buildings and tunnels [11], and thus this kind of method will be invalid in many cases.

Visual Odometry. Visual odometry is a method to determine the position and orientation of a robot by analyzing the associated camera images [12]. It is a localization method based on computer vision. Lots of methods adopt camera in trajectory reconstruction. The method in [13] locates two surveillance cameras and simultaneously reconstructs object's 3D trajectories. Silvatti et al. utilizes submerged video cameras in an underwater 3D motion capture system to reconstruct 3D trajectory [14]. Nevertheless, computer vision methods are not effective in textureless environment usually, such as wide snowfield or dark night.

Wireless-Based Localization. Wireless-based localization is mainly applied in indoor situations. There are many wireless devices. The most widely-used wireless devices are Wi-Fi and Bluetooth [15,16]. A most common wireless localization technique is called "fingerprinting" for getting position information [17]. One can calculate its location according to the magnitude of received signals from several base stations. The inconvenience of these wireless approaches is that they require base stations set up in advance in the scene. Hence, the Wi-Fi/Bluetooth signal is not always available for most common situations.

Ultrasonic Sensor/LIDAR-Based Localization. The principles of localization through LIDAR [18] and ultrasonic sensors [19] are similar. They measure the distance to a certain target by calculating time between when emit a pulse and when receive echoes. Nonetheless, LIDAR is so expensive that most mobile devices are not equipped with it, while the measuring range of ultrasonic sensors is too limited to cover most area [20].

Inertial Navigation. Inertial navigation is to get one's location through IMU (inertial measurement unit). IMU is a combination of 3-axis accelerometers and 3-axis gyroscopes. It can measure the specific force and angular velocity of an object. According to the work of Titterton and Weston [21], trajectory can be reconstructed through the second-order integration of the accelerometer signals. The attitude information could be obtained through gyroscopes, which assists the accelerometer signals to be transformed from the body frame to the inertial frame [22].

Some work studies how to utilize the IMU signals to reconstruct trajectory [4]. For example, writing trajectory based on IMU is reconstructed through a pen tip direction estimation method in the work of Toyozumi et al. [23]. An error compensation method and a multi-axis dynamic switch developed by Wang et al. to minimize the cumulative errors caused by sensors [24].

Because of the merits of MEMS IMU, such as low-cost, light weight and small size, most mobile devices are equipped with MEMS IMU. But the main disadvantage of the MEMS IMU is its low accuracy on account of errors [25].

Taking into account the advantages and disadvantages, MEMS IMU signals are employed as a main source of data for reconstructing trajectory in our work. IMUs depend less on the environment. IMU-based trajectory reconstruction methods are quite practicle for MEMS IMU is commonly equipped in mobile devices. However, it is impossible to reconstruct trajectory by IMU alone due to the errors carried in MEMS IMU. Therefore, we propose a sensor-fusion-based method to eliminate errors in MEMS IMU signals.

2.2 Methods to Calibrate IMU Signals

Recent advances in MEMS (Micro-Electro-Mechanical Systems) technique bring possibility of producing small and light inertial navigation systems. On the other hand, the main problem of MEMS devices is its low accuracy. The error, as illustrated in Fig. 1, is indicated by bias and noise in their measurements as elaborated in the work of Woodman [5]. During trajectory reconstruction, the accelerometer signals are integrated twice, which makes the error grow even rapidly.

Many researchers study on reducing errors caused by IMU devices. A zero velocity compensation (ZVC) mechanism to reduce the accumulative errors of IMUs proposed by Yang et al. [6]. Pedley applies linear least squares optimization to compute the recalibration parameters from the available measurements to reduce errors [7].

Some other methods adopt Kalman Filter to combine IMU with computer vision to improve accuracy. For instance, a VISINAV algorithm is presented to enable planetary landing, which utilizes an extended Kalman filter (EKF)

to reduce errors [26]. Additionally, a state space model is applied to estimate the navigation states in an extended Kalman filter [8]. However, the error-state vector, which is estimated in advance, has a direct impact on the result, that is, large deviation of error-state estimation leads to poor results.

Most of the works mentioned before do not aim at low-cost MEMS devices which carry much more errors, while low-cost MEMS devices are commonly used because of convenience.

Consequently, we focus on the trajectory reconstruction from low-cost MEMS IMU in this paper. A type of quadrotor drone is taken as an example of mobile devices. During the course, different error models are designed for different types of errors, so that diverse errors can be eliminated in targeted ways.

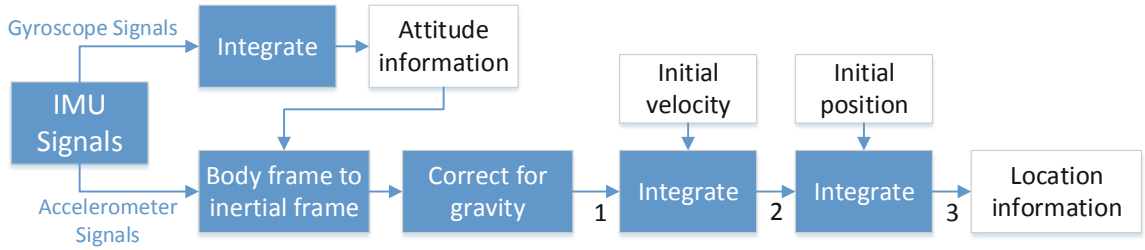

Fig. 2. The principle of trajectory reconstruction based on IMU. In the figure, 1. Accelerometer data in inertial frame; 2. Velocity calculated from accelerometer data; 3. Location information calculated from velocity.

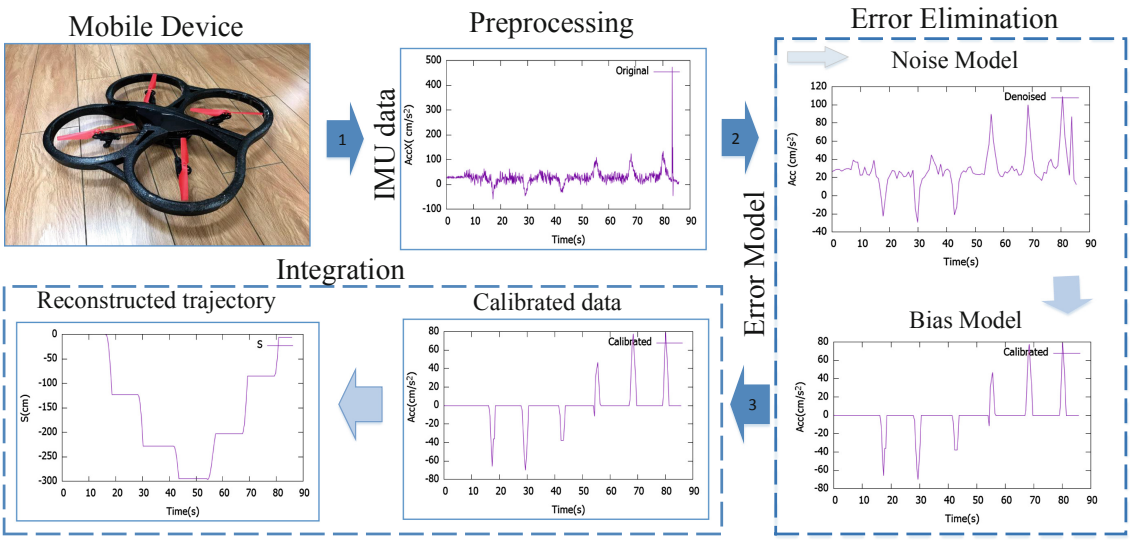

Fig. 3. The pipeline of our method: (1) Data collection and preprocessing; (2) Error elimination according to error model; (3) Integration and trajectory reconstruction.

3 IMU-Based Trajectory Reconstruction

Trajectory can be reconstructed from IMU signals shown in Fig. 2 in ideal status. The measured accelerometer data is transferred from the body frame to

the inertial frame according to the attitude estimation calculated by gyroscope signals. The accelerometer data is then corrected for gravity. Next, velocity is obtained from the integration of the accelerometer data. Finally, trajectory is reconstructed from the integration of the velocity, that is, the second-order integration of the accelerometer data [5].

However, inevitable errors exist in MEMS IMU signals. That's why the trajectory reconstruction cannot be directly calculated in such ideal way.

Thus, we propose a trajectory reconstruction method for quadrotor drones in this paper, which utilizes the measurement of IMU and other sensors. The pipeline of our method is illustrated in Fig. 3. It consists of three phases:

1. Sensor data collection and preprocess;
2. Error elimination on the basis of our error model;
3. Trajectory reconstruction through integration of calibrated accelerometer data.

First, sensor data, which mainly includes the accelerometer, gyroscope and ultrasonic signals, is collected discretely from the target quadrotor drones. Trajectory is reconstructed in the inertial frame while IMU data is collected in the body frame [27], so a coordinate transformation is needed to perform in a preprocessing phase.

We denotes the raw measured accelerometer data at time point t in the body frame as $a^0(t) = (a_x^0(t), a_y^0(t), a_z^0(t))^T$, and its correspondence in the inertial frame as $\tilde{a}(t) = (\tilde{a}_x(t), \tilde{a}_y(t), \tilde{a}_z(t))^T$. Thus, the coordinate transformation can be formulated as

$$a^0(t) = R(\phi, \theta, \varphi) \cdot \tilde{a}(t), \tag{1}$$

where $R(\phi, \theta, \varphi)$ is the rotation matrix from the inertial frame to the body frame [28]; ϕ , θ and φ stand for the three Euler angles between the two frames.

Because the raw IMU signals contain a lot of errors due to the low accuracy of MEMS, the most important step in our pipeline is to eliminate those errors before the data is used for trajectory calculation.

As mentioned above, errors in MEMS are comprised of the noise and the bias. Different from traditional methods which process the two parts together, we divide the error model into a noise component $\epsilon_n(t)$ and a bias component $\epsilon_b(t)$, and process them separately in the second step. So we design the error model as,

$$\tilde{a}(t) = \alpha \cdot a(t) + \epsilon_n(t) + \epsilon_b(t) - H \cdot g, \tag{2}$$

where $a(t)$ is the calibrated accelerometer data, α is a scale factor between the measured inertial data and the actual data, $H = (0, 0, 1)^T$, and g stands for the gravitational acceleration.

The two parts are processed separately in the next phase with accordance to our error model. We first reduce the noise, then eliminate the bias. Thus, a set of calibrated accelerometer data $a(t)$ is obtained.

Finally, the calibrated accelerometer data is integrated over time to obtain the 3D trajectory. Hence, the 3D position at time t_i, noted as $S_i = (S_{ix}, S_{iy}, S_{iz})^T$, which is calculated as follows:

$$S_i = V_i * \Delta t_i + S_{i-1} = \sum_{k=0}^{i} v_k * \Delta t_k = \sum_{k=0}^{i} (\sum_{j=0}^{k} a(t_j)\Delta t_j)\Delta t_k, \tag{3}$$

where $a(t_i)$ represents the ith signal of the calibrated accelerometer data, and Δt_i is the time interval between t_i and t_{i-1}. V_i stands for the velocity calculated from $a(t)$. Thence, $\{S_i | i = 1, 2, ..., n\}$ composes the reconstructed trajectory.

4 Reducing Noise

According to the previous analysis, the errors can be classified into two types, noise and bias. As shown in Fig. 1, the measured accelerometer signals seriously oscillate at a large amplitude around certain values (marked by red lines). This serious vibration results in noise. On the other hand, even when the device remains still while its signals are being collected, the red line keeps drifting from its true value, and presents a step shaped line instead of a straight line. This kind of data drifting is called bias.

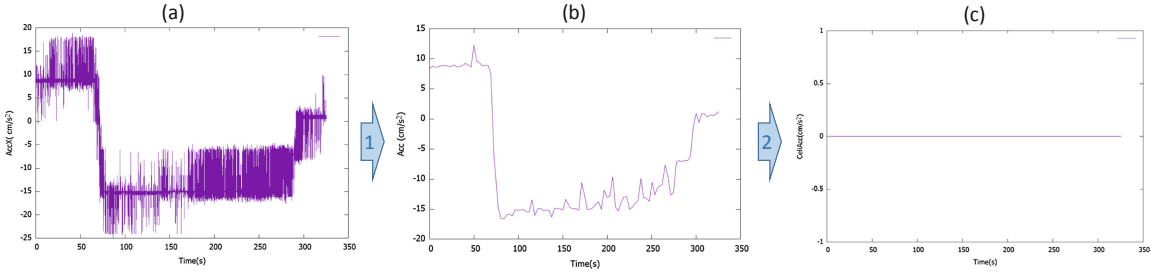

Fig. 4. Eliminating noise and bias errors in the accelerometer data. (a) Raw measured accelerometer signals collected from a static devices; (b) After reducing noise; (c) After eliminating bias [30].

Here, in this section, we focus on how to reduce noise, and the other type of the errors will be discussed in the following section.

The noise in the MEMS IMU data could be regarded as a high frequency signal superimposed on the real signal, while the real signal is in low frequency. Therefore, valid data could be obtained by filtering out the noise through a low-pass filter.

Since the measured data is in the discrete form, the low-pass filter can be designed through FIR (finte impulse response filter) [29].

Hence, given the measured raw accelerometer signals $\{\tilde{a}(t_i) | i = 1, 2, ..., n\}$, the denoised accelerometer data $\{\hat{a}'(t_i)\}$ is filtered out by

$$\hat{a}'(t_i) = \sum_{k=0}^{n} h(t_k)\tilde{a}(t_i - t_k), \tag{4}$$

where $h(.)$ is the impulse response function of the low-pass filter. The filter is

presented in a convolution form in the time domain. Here we adopt Rectangular as the low-pass window function. The parameters in Rectangular will be adjusted to a proper value.

In order to achieve a better denoised result, a down sampling is applied to the filtered data, which could reduce the amount of the following calculation as well. Thus, the final denoised accelerometer data set $\{\hat{a}(t)\}$ is a subset of $\{\hat{a}'(t)\}$, which is down sampled at a certain period δ. In our current implementation, we adopt $\delta = 100$ ms.

In fact, the high-frequency noise also exists in the signals of other sensors, such as gyroscopes and ultrasonic sensors, and thus the data of these sensors should also be denoised in the similar way. After denoising, the smoother result will be used in the following bias elimination method.

5 Eliminating Bias

Figure 4(b) shows that, a relatively smooth curve of the accelerometer data is obtained after filtering the noise. But the value is not correct yet for bias errors still exist. It is represented as the data drifting. It is varying over time in low frequency [5], as demonstrated by the change of the red lines in Fig. 1. In previous works, the bias is removed only once before the whole movement, which ignores the dynamical bias. Hence we are aiming to improve this by dynamical bias elimination during the movement.

Through the observation of the details in the denoised accelerometer data, we found that it is difficult to determine when the IMU produces a bias by only analyzing the absolute value of accelerometer data. Therefore, the moment when a bias happens should be found out, in order to eliminate bias correctly and dynamically during the movement of the device.

Although bias may occur on all MEMS sensors aperiodically, it is less probably to occur on multiple sensors at the same time. Hence, a sensor-fusion-based method is proposed to detect the moment when bias occurs.

The denoised accelerometer data can be segmented into a series of sections along the timeline, according to these moments. In each section, we consider the device maintaining the same motion status, which means the accelerometer value should be a constant.

Here, we define each section on the timeline as an *event*, and the beginning moment of each *event* is called an *event timestamp*.

We then take different strategies to eliminate bias errors according to different *event* tags. This method, through segmentation on the timeline, will effectively compensate for the accumulation of bias errors over time.

5.1 Event Detection

For the sake of event timestamps detection, we first inspect the derivative of the accelerometer data over time, which indicates the vibration of the accelerometer data. If the absolute derivative value is greater than a certain positive threshold

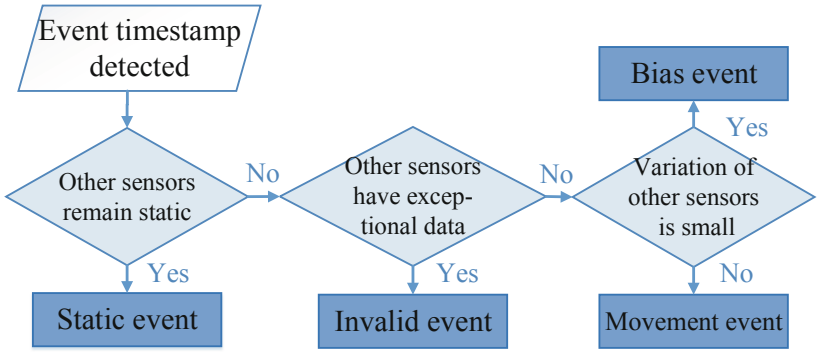

Fig. 5. Determining the type of an event according to the status of multiple sensors [30].

τ, it indicates the status of the IMU is being changed, i.e., an event is happening. Hence, this particular moment is recorded as an event timestamp, the beginning of a new event.

However, the initially detected events are not necessarily the bias events that aimed to process. Maybe the event timestamp is when a real movement happens. Sometimes, exception events will also occur. Because the possibility of bias occurring simultaneously in multiple sensors is extremely low, we refer to the status of multiple different sensors as an assistant, e.g. gyroscopes and ultrasonic sensors. Therefore, different types of the events are discriminated through this way.

By analyzing the data status from other sensors, initial events can be classified into the following four types, as illustrated in Fig. 5:

Bias Event. Bias event is when the device produces bias errors. The accelerometer data in this event needs to be corrected. If the variation of other sensors is small at the event timestamp while the event still happen, the bias is considered to happen. For instance, the accelerometer data has an intense change while other sensors data remains stable, which indicates bias occurs to the accelerometer sensor. Therefore, the event is marked as a bias event.

Movement Event. This type of event indicates that the device is in a movement, that is, the device is moving in a certain direction. In this case, the change of the accelerometer data is caused by a real movement, and other sensors data should correspondingly show reasonable variations. For instance, the accelerometer data has an intense change while other sensors data also has a coordinated change in the correspond direction, which indicates the device is taking an action. Therefore, the event is marked as a movement event.

Static Event. In this case, the mobile device is actually in a static status, i.e., its accelerometer data and speed should be zero. That can be deduced by Euler angles (attitude angles: picth, roll, yaw) ϕ, θ and φ of the device. If the values of the picth, roll and the change of yaw are all very near to zero, and the status lasts a period time with the data of other sensors also has little variation, the event is regarded as a static event. Additionally, the static event should last for a little while.

Invalid Event. In some special cases, we may encounter invalid data occasionally. For instance, when the value exceeds the measuring range, or the data is in a violent shaking, those cases cannot present the real status. Therefore, if other sensors exhibit an irregular status, e.g. the gyroscope data vibrates frequently and severely in a very short period, the event is marked as an invalid event.

Here, bias event and movement event are regarded as regular types, while static event and invalid event are considered as handful types, which happens occasionally. Besides, we may design that some static events happen on purpose in order to eliminate cumulative errors.

5.2 Processing Algorithm

The accelerometer data can be segmented into events after the detection of event timestamps. Then, the accelorometer data is calibrated in each event, according to the type of the event. Consequently, bias errors will all be eliminated. The bias elimination algorithm is listed as Algorithm 1.

Here, we denote the calibrated accelerometer value in the previous section as $PreAcc$, and the bias value of current section as $BiasValue$, both initialized as zero.

If the event corresponds to a bias event, the motion status of the device is not actually changing although a data drifting is occurring. Hence, we correct the accelerometer data in this event to the calibrated data in the previous section (Fig. 6(b)). Meanwhile, the bias value, i.e., the difference between the measured data and the calibrated data, is updated and recorded as $BiasValue$. It will be applied in the processing of the subsequent events.

If the event corresponds to a movement event, the variation of the accelerometer data is caused by a real movement. Then, the calibrated accelerometer value can be calculated as the median of the data in this event subtracting current recorded $BiasValue$ (Fig. 5(c)). After that, $PreAcc$ is updated as the same value for the calculation of the following events.

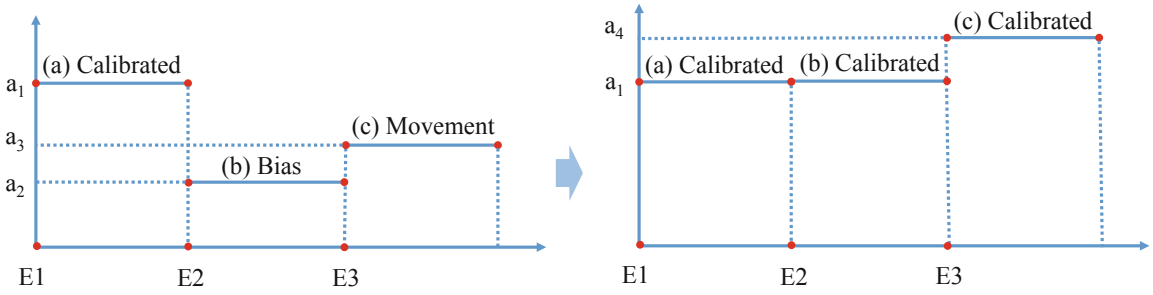

Fig. 6. An example of event processing [30]. Accelerometer data is calibrated by sections according to different event types.

Algorithm 1. Bias Elimination Algorithm.

Input:

1. Denoised accelerometer data $\{\hat{a}(t)|t = 1, 2, ..., n\}$;
2. Detected events $\{E_i|i = 1, 2, ..., m\}$;

Output:

1. Calibrated accelerometer data $\{a(t_i)|i = 1, 2, .., n\}$;

Definition:

1. t_i: the event timestamp of E_i.
2. $median(.)$: a function returns the median value of a data set.
3. $BiasValue$: the bias value of the accelerometer data, initialized as zero;
4. $PreAcc$: the calibrated accelerometer data of the prevoius event, initialized as zero;

Algorithm:

for i from 1 to m **do**
  if $E_i == static$ **then**
    while $t \in [t_i, t_{i+1})$ **do**
      $a(t) = 0.0$
    end while
    $PreAcc = 0.0$
    $BiasValue = median(\hat{a}(t), t \in [t_i, t_{i+1}))$
  else if $E_i == invalid$ **then**
    All parameters remain unchanged.
  else if $E_i == bias$ **then**
    while $t \in [t_i, t_{i+1})$ **do**
      $a(t) = PreAcc$
    end while
    $curAcc = median(\hat{a}(t), t \in [t_i, t_{i+1}))$
    $BiasValue = curAcc - PreAcc$
  else if $E_i == movement$ **then**
    $curAcc = median(\hat{a}(t), t \in [t_i, t_{i+1}))$
    while $t \in [t_i, t_{i+1})$ **do**
      $a(t) = curAcc - BiasValue$
    end while
    $PreAcc = curAcc - BiasValue$
  end if
end for

In the case of a static event, the device stays still and motionless. Hence, the accelerometer data in this event should be reset to zero. $PreAcc$ is also cleared to zero, and the median of the accelerometer data in this event is recorded as the $BiasValue$.

Finally, for an invalid event, it can be view as the same as the previous event because of the very short time. Thus, the process of this event follows the previous event, and all parameters remain unchanged.

Therefore, the output of the algorithm is the final calibrated accelerometer data, which could be directly applied in trajectory reconstruction.

Table 1. Onboard sensors of AR. Drone 2.0 [30].

Sensors	Specifications
3-axis accelerometers	Bosch BMA 150, Measuring range: ± 2g
2-axis gyroscopes	Invensense IDG500, Measuring rate: up to 500 deg/s
1-axis gyroscope	Epson XV3700, On vertical axis
Ultrasonic sensor	Measuring rate: 25 Hz
Vertical camera	64° diagonal lens, Frame rate: 60 fps
Front camera	93° wide-angle diagonal lens, Frame rate: 15 fps

6 Experiments

We have proposed trajectory reconstruction method for quadrotor drones so far. The data from the accelerometer and other sensors on the device is utilized in the reconstruction in a manner of sensor-fusion. In order to validate the effectiveness of our method, we apply it to several quadrotor drones with the same type, which are equipped with low-cost MEMS IMU and other sensors.

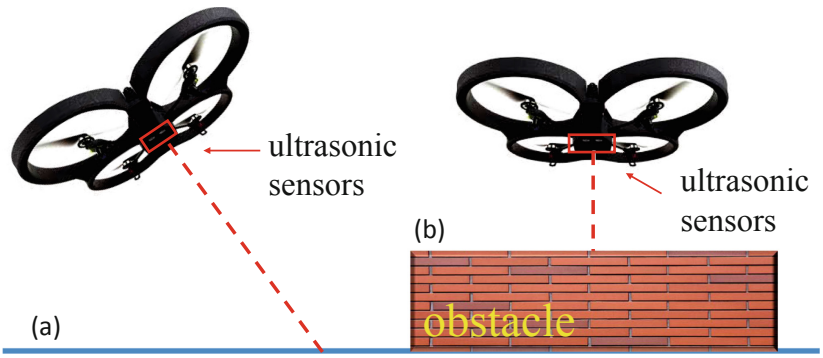

Fig. 7. The ultrasonic measurement may deviate from the actual height during movement or flying over obstacles [30]. (The red rectangels stand for the ultrasonic sensor on the drone, and the red dotted lines indicate the ultrasonic measurements). (Color figure online)

6.1 Implementations

In our experiments, we adopt the Parrot AR. Drone 2.0 as the target quadrotor drones. AR. Drone 2.0 is a lightweight quadrotor. A Linux based real-time operating system and multiple onboard sensors are equipped on it. The sensors and their specifications are listed in Table 1.

Accelerometers provide the major data for the calculation of the trajectory among all these sensors, while the others are used as auxiliary sensors in the bias elimination method.

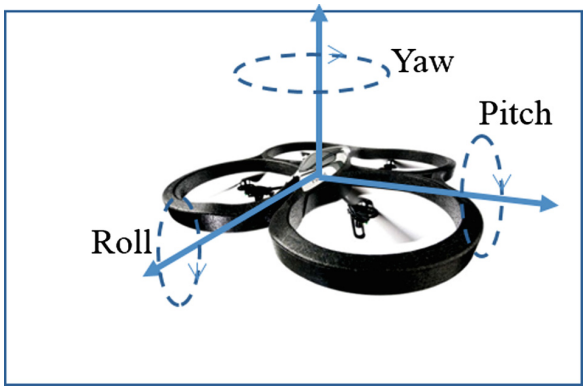

Fig. 8. Euler angles on quadrotor drones.

The gyroscope data is applied for attitude estimation, i.e., calculate the euler angel: pitch, roll and yaw (shown in Fig. 8). It can also be used for event timestamp detection on the X or Y axis, because the drone would tilt if there is a movement on the X - Y plane, and that will result in a variation of the gyroscope measurement.

There is an ultrasonic sensor on the bottom of the AR. Drone. It measures the distance from the drone to the ground. The absolute derivative of the ultrasonic data is utilized in event detection on the Z-axis. However, it is not directly used as the trajectory on the Z-axis, because when the drone tilts during its movements, the angle of the ultrasonic sensor would also change. Therefore, its measurement can not reflect the actual height of the drone (Fig. 7(a)). Besides, when the drone flies over a series of obstacles, large variations may also occur in its measurement (Fig. 7(b)).

In addition, we perform our experiments indoor for the ultrasonic sensor has a limited range. Indoor experiments can also simplify the flight condition, like the absence of wind. On the other hand, indoor localization is more difficult for there is no GPS signals indoors.

6.2 Experimental Environment

We apply our experiments indoors, where there is no GPS signals for reference. Moreover, indoor environments may avoid the influence of winds and other

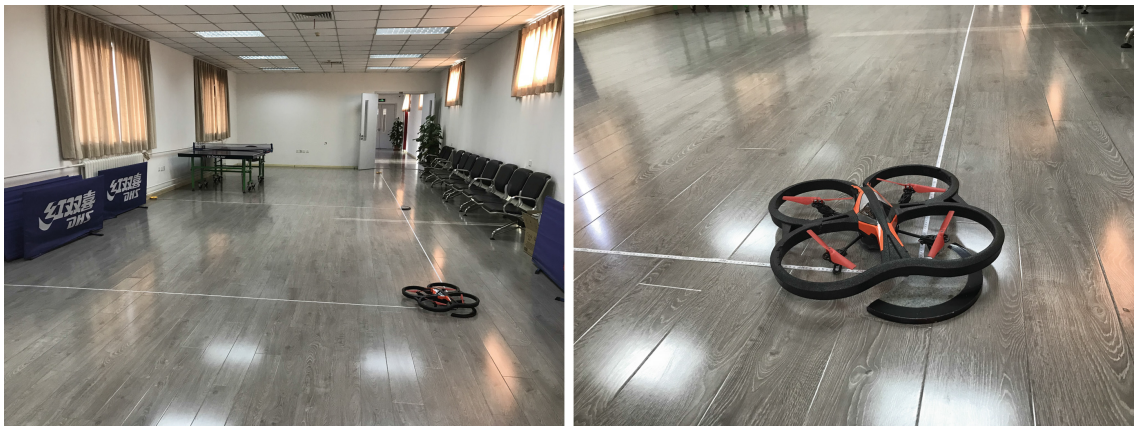

Fig. 9. Our experimental environment.

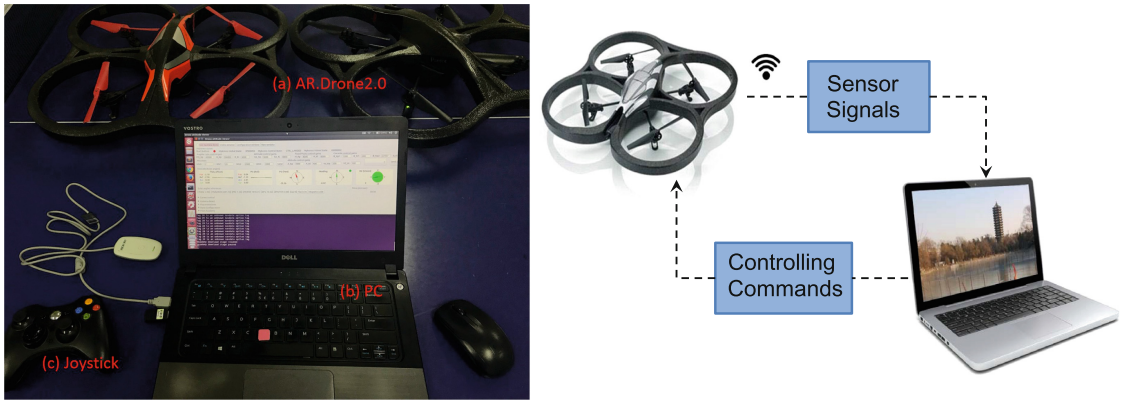

Fig. 10. Drone Control Technique.

affects. The experimental environment is shown in Fig. 9 left, where the floor is relatively flat. Note that surfaces of obstacles in the environment are relatively flat as well, so that the ultrasonic sensor can work effectively as an assist of accelerometer data in Z-axis.

Flight Route Control

As shown in Fig. 10, the Parrot AR.Drone 2.0 communicates with PC through Wi-Fi. We control the automatic flight route of the drone through its SDK[1]. A joystick is also used to avoid emergencies, e.g., that the drone hits the wall or breaks the window. The controlling commands of the joystick are sent to the drone by means of PC.

Trjactory Groud-Truth Measurement

The ground-truth of our experiment is measured in a manual way. Several rulers are fixed in the environment, as shown in Fig. 11. At the same time, a laser range finder is applied to sample the distance between the drone and the ground or between the drone and the staring position (shown in Fig. 9(b)). Since the flight route is determined by programming, we could approximate the movement of the

[1] http://developer.parrot.com/products.html.

drone, which makes our measurement easier. Besides, because the drone itself may be influenced by the environment, such as the temperature and its battery power, the drone cannot be controlled accurately. That's why the drone routes do not perfectly match the designed route.

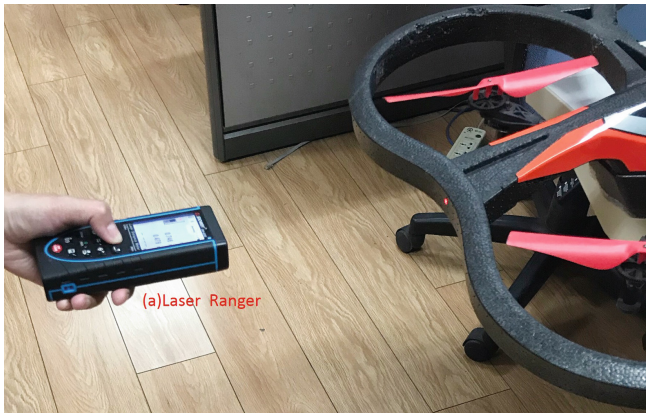

Fig. 11. Measure the distance by a laser ranger.

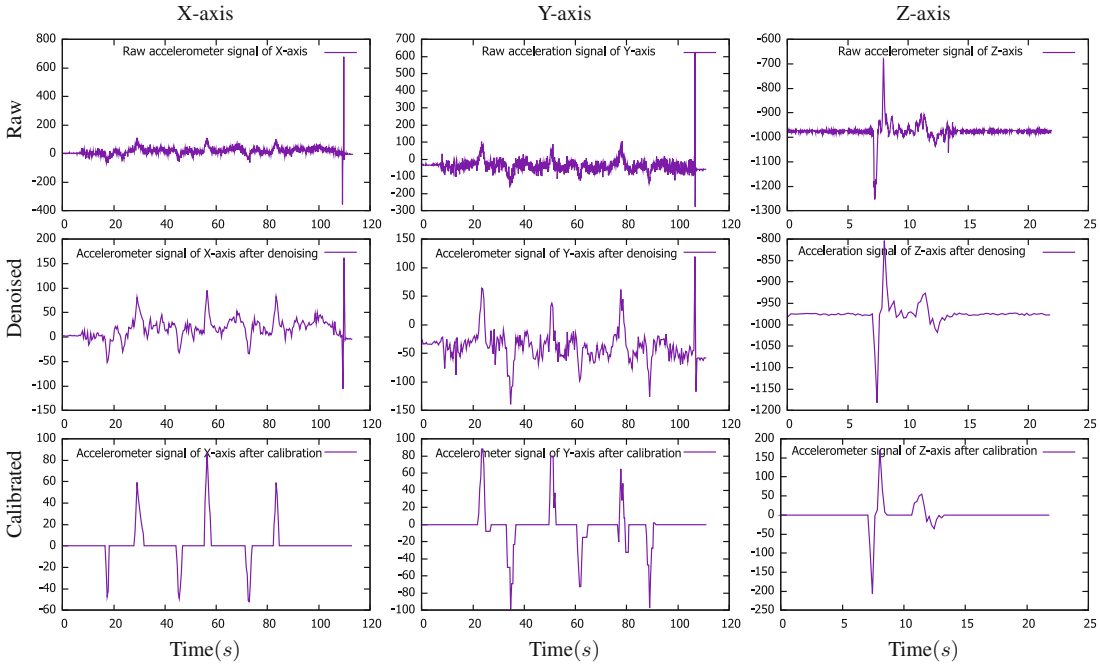

Fig. 12. The result of error elimination for the accelerometer data on three axes [30]. The first row is the raw accelerometer data collected from a quadrotor drone. The second row shows the signal after denoising by a low-pass filter and down sampling. The last row is the final calibrated output of accelerometer data after bias elimination.

6.3 Experimental Results

A large amount of experiments are performed in order to verify the effectiveness of the proposed algorithm. Here are some experimental results demonstrated in this section.

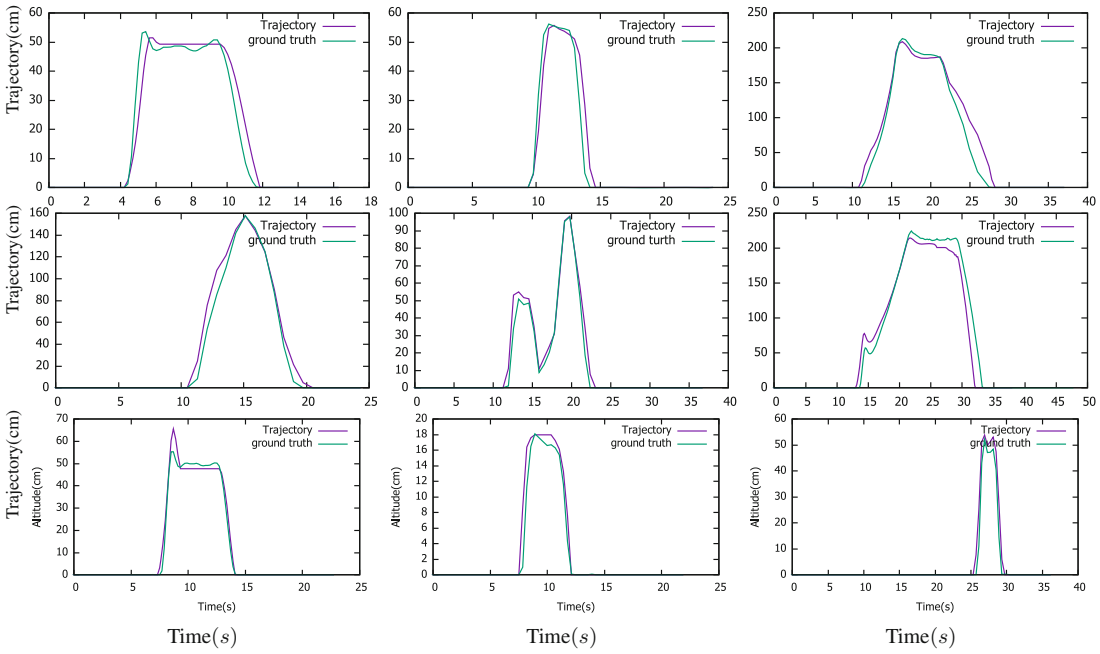

Fig. 13. Trajectory reconstruction results on a single-axis (purple lines), comparing with the ground truth (green lines). (Color figure online)

Results of Error Elimination. As shown in Fig. 12, the first row is the raw accelerometer data. It seems to be totally out of order because of too much noise and bias. Thus, it is impossible to reconstruct the trajectory through these raw signals. In the second row, the signals are filtered out through a low-pass filter with down sampling, which becomes smoother after denoising. But bias errors still exist in there. In the last row, it is the final calibrated accelerometer data after bias elimination. Hence, valid signals are finally extracted by our method after redundant error signals are removed and the outliers filter away.

Trajectory Reconstruction for Single-Axis Movements. We first test our trajectory reconstruction method in relatively simple direction. We allow the drone to move in only one direction along X, Y, or Z axis. At the same time, we keep it invariant in the other two directions. Therefore, the data and the motion status is on only one axis through inspection. We assume the accelerometer data on the other two axes are always zero. As shown in Fig. 13, the purple lines are the reconstructed trajectories by our method, and the green lines are the ground truth trajectory. Through the comparison, we can see that our result is close to the actual movement. The errors of reconstruction are controlled within 10 cm in each axis.

Algorithm Calibration. In order to obtain better results, the event detection thresholds τ for the accelerometer or other sensors' data, together with the window size of low-pass filter, need to be adjusted to a proper value. In fact, this parameters adjustment of algorithm can be adaptively accomplished in our method. We first pick a small part of the data at the beginning of the flight, and

obtain the optimal thresholds interactively. Then, the rest of the trajectory can be automatically reconstructed with these thresholds.

An example is given in Fig. 14, only the red part of the result is the one after the algorithm calibration is accomplished interactively. The purple line is the result when the parameters are fixed before. The final result is similar with what we designed in advance, which shows the adaptability of our method.

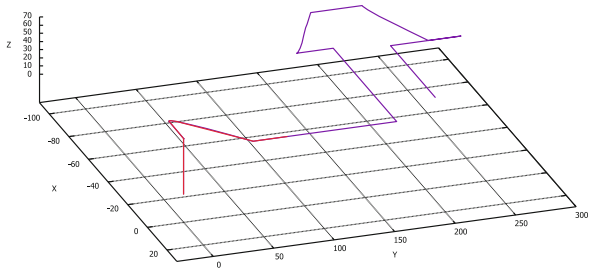

Fig. 14. Algorithm calibration [30]. Optimal parameters are interactively searched for using the first part of data (red line), and then the whole trajectory (purple line) can be automatically reconstructed. (Color figure online)

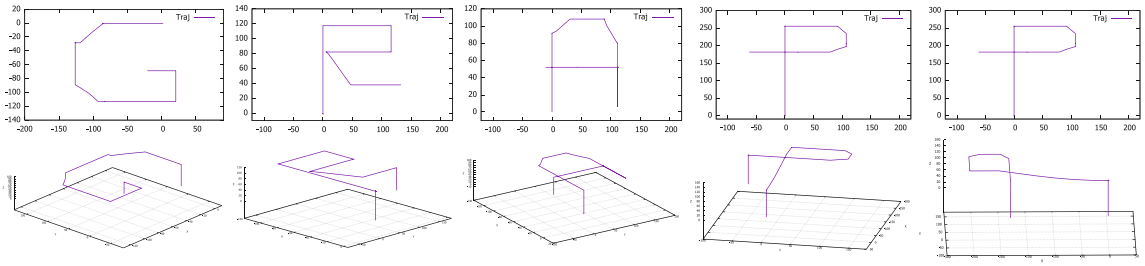

Fig. 15. Trajectories reconstructed from Ar.drone. We design target trajectories as the letters of "GRAPP". And the results match the targets well. The first row is the trajectory in X-Y plane. The second row is each corresponding one in three-dimensional space [30].

Trajectory Reconstruction for Multiple-Axis Movements. After the success of single-axis tests, we carry out more complicated experiments of reconstructing trajectory on multiple axes. Drones fly along given routes with various shapes. The drone is controlled by PC through Wi-Fi. The route is designed in advance by a program so that it can fly at a relatively constant speed, and fly straightly in the given directions.

Several groups of reconstruction results are given in Figs. 15, 16, 17 and 18. Target trajectories are designed as meaningful letters. We can see that, the reconstructed routes are close to what we designed. After denoising and bias elimination, the valid accelerometer data can be extracted from the raw signal. The 3D trajectories of the drone can be correctly reconstructed. The reconstructed trajectories (purple lines) coincide with the ground truth routes (green lines).

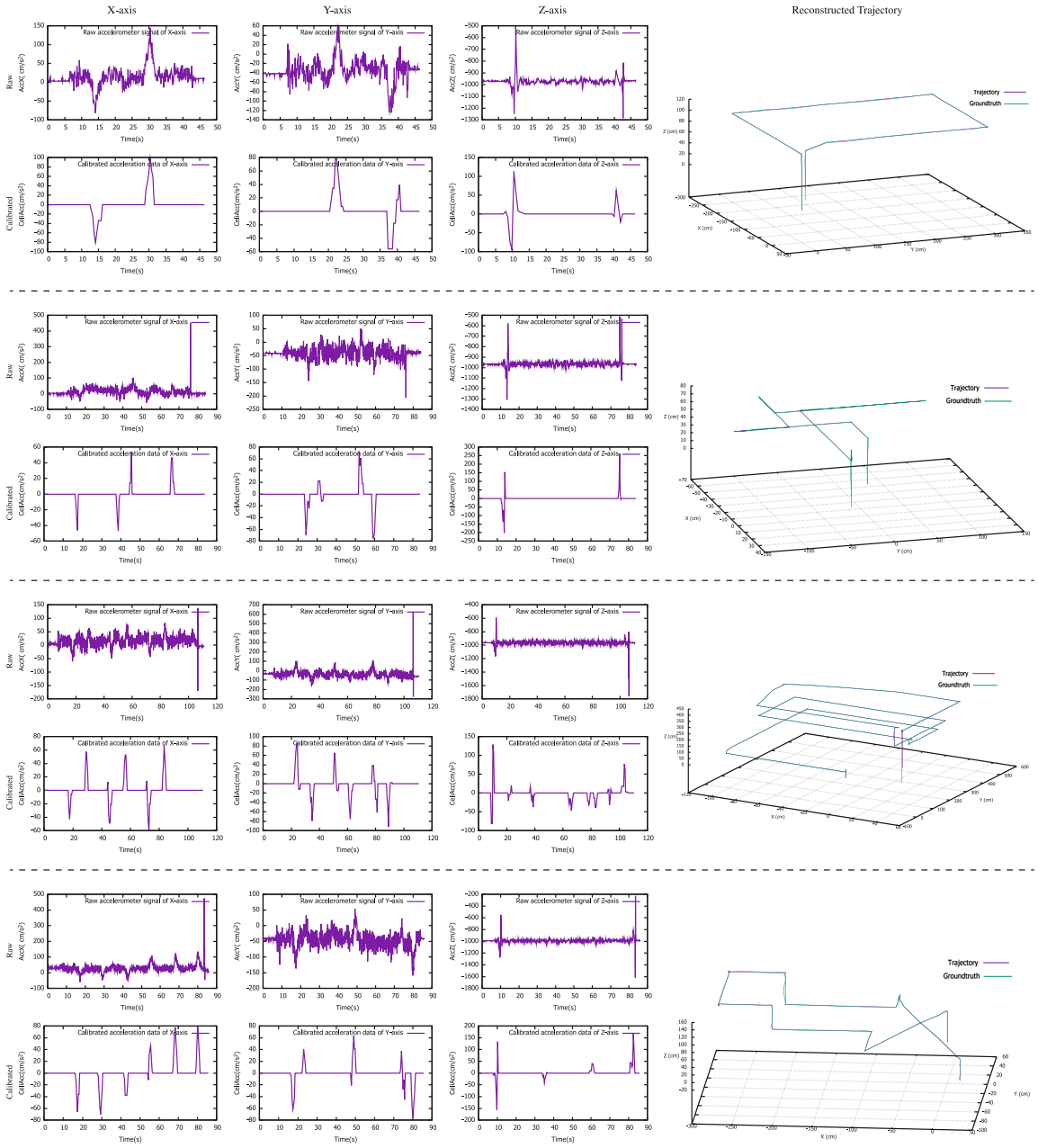

Fig. 16. Multi-axis trajectory reconstruction results. In each group, the left three columns show the raw and calibrated accelerometer data on X, Y and Z axis. The last column illustrates the final 3D reconstructed trajectories of the drone (purple line), comparing with the ground truth (green line). (Color figure online)

However, due to the instability of the controlling algorithm inside the drone, the actual flying route of the drone may have a little slight offsets. The offsets are too small to be detected due to the low accuracy of onboard sensors, hence they would be ignored by our algorithm. It is the reason that reconstructed trajectory is a little smoother and straighter than the actual trajectory.

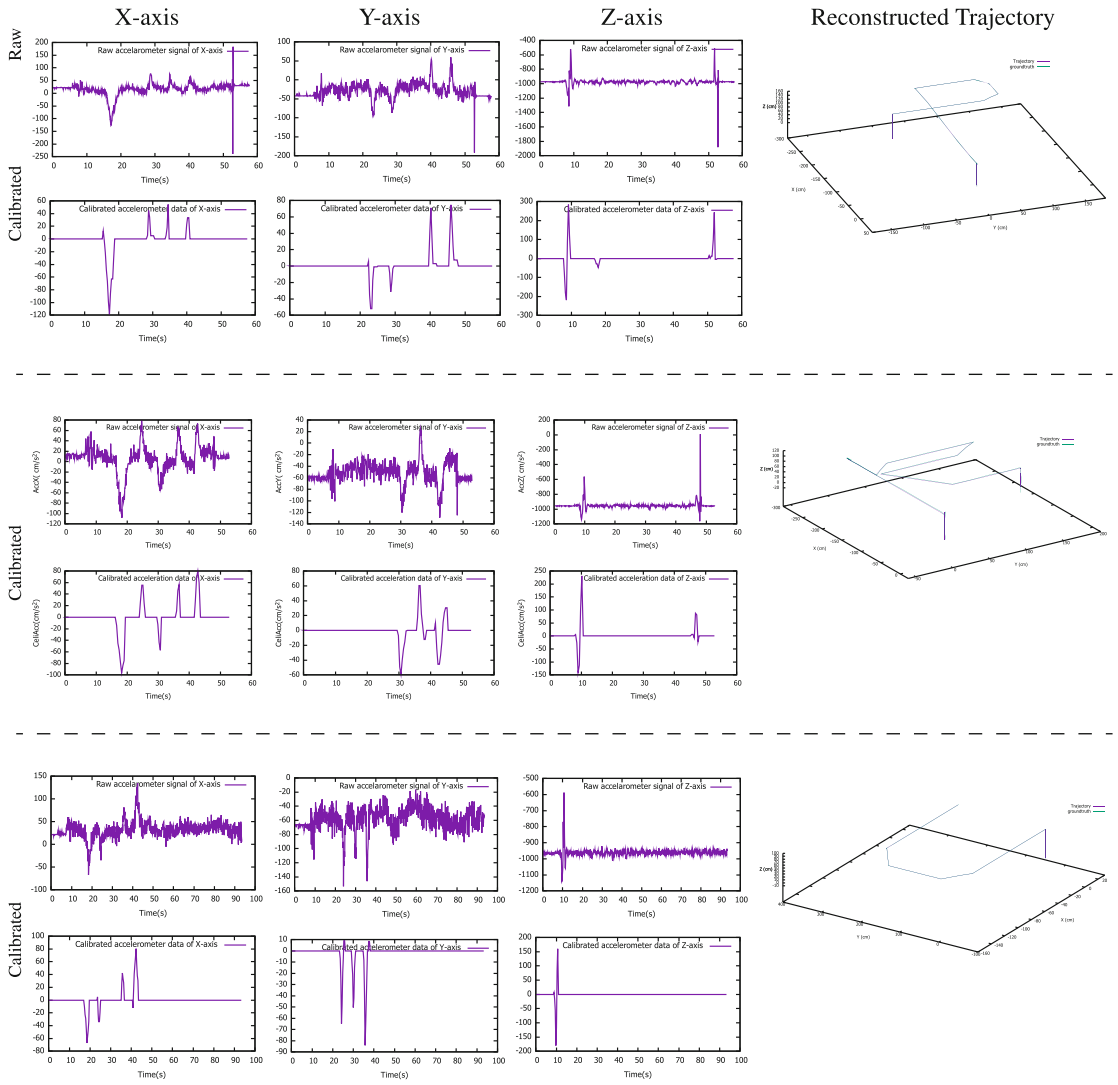

Fig. 17. Reconstructed trajectory "PKU", which is the abbreviation of our school. In each group, the left three columns show the raw and calibrated accelerometer data on X, Y and Z axis. The last column illustrates the final 3D reconstructed trajectories of the drone (purple line), comparing with the ground truth (green line). (Color figure online)

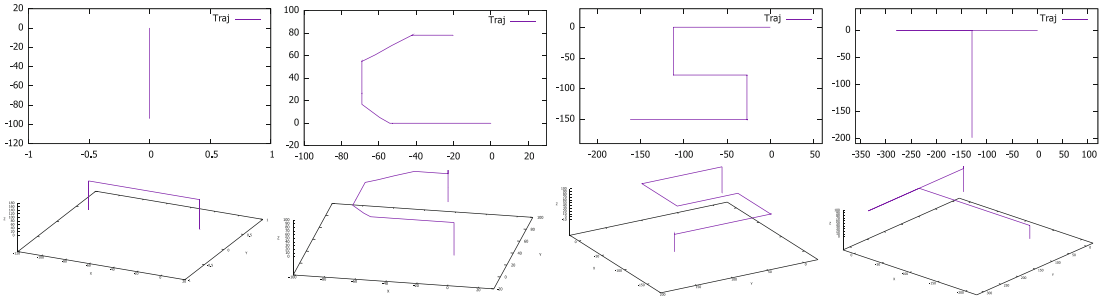

Fig. 18. Reconstructed trajectory "ICST", which is the abbreviation of our institute.

7 Conclusions

We present a novel method for trajectory reconstruction of quadrotor drones, which suffers from unavoidable errors of low-cost MEMS IMUs. There are two types of errors in MEMS IMUs: noise and bias. In our method, they are processed separately, according to their different characteristics. A low-pass filter with downsampling is applied to reduce the noise, and then a sensor-fusion-based algorithm is applied to dynamically eliminate the bias. Therefore, the trajectory is reconstructed based on the calibrated data.

In fact, this sensor-fusion-based bias elimination method can be extended to employ various kinds of sensors, such as cameras, gradienters, magnetometers, etc. That makes the method more practical. Theoretically, the trajectory reconstruction method can also be applied to any mobile device as long as it is equipped with IMUs.

In our current implementation, a rugged environment may cause failure in reconstruction. That is because the ultrasonic senor, as one of auxiliary sensors, would become invalid in such cases. In our future work, we will try to enroll more auxiliary sensors to handle more complicated conditions, so that our method may be applied to more scenarios. Besides, our method performs better in reconstructing straight lines than in curves due to the low accuracy of sensors. To solve this problem, we can approximate the curves with a set of line segments.

On another aspect, the flight routes of the drone do not perfectly match what we design, for the environment and battery power will influence the flying status of drone. Thus, we will put emphasis on the route controlling module as well, in order to make the drone fly more steadily.

Since the trajectory can provide important viewpoint information, our trajectory reconstruction method can be applied in various applications, for instance, automatic drone navigation, 3D reconstruction, map building and stereoscopic video synthesizing.

Acknowledgements. This work is partially supported by National Key Research and Development Program of China (2016QY02D0304) and NSFC grants (61602012).

References

1. Kopf, J., Cohen, M.F., Szeliski, R.: First-person hyper-lapse videos. ACM Trans. Graph. (TOG) **33**, 78 (2014)
2. Ten Hagen, S., Krose, B.: Trajectory reconstruction for self-localization and map building. In: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2002, vol. 2, pp. 1796–1801 (2002)
3. Nägeli, T., Meier, L., Domahidi, A., Alonso-Mora, J., Hilliges, O.: Real-time planning for automated multi-view drone cinematography. ACM Trans. Graph. (TOG) **36**, 132 (2017)
4. Suvorova, S., Vaithianathan, T., Caelli, T.: Action trajectory reconstruction from inertial sensor measurements. In: 2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA), pp. 989–994 (2012)

5. Woodman, O.J.: An introduction to inertial navigation. Technical report, University of Cambridge, Computer Laboratory (2007)
6. Yang, J., et al.: Analysis and compensation of errors in the input device based on inertial sensors. In: Proceedings of the International Conference on Information Technology: Coding and Computing, ITCC 2004, vol. 2, pp. 790–796 (2004)
7. Pedley, M.: High precision calibration of a three-axis accelerometer. Freescale Semicond. Appl. Note **1** (2013)
8. Fredrikstad, T.E.N.: Vision aided inertial navigation. Master's thesis, NTNU (2016)
9. Kee, C., Parkinson, B.W., Axelrad, P.: Wide area differential GPS. Navigation **38**, 123–145 (1991)
10. Farrell, J.: Aided navigation: GPS with high rate sensors (2008)
11. Kleusberg, A., Langley, R.B.: The limitations of GPS. GPS World **1** (1990)
12. Huang, A.S., et al.: Visual odometry and mapping for autonomous flight using an RGB-D camera. In: Christensen, Henrik I., Khatib, Oussama (eds.) Robotics Research. STAR, vol. 100, pp. 235–252. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-29363-9_14
13. Pflugfelder, R., Bischof, H.: Localization and trajectory reconstruction in surveillance cameras with nonoverlapping views. IEEE Trans. Pattern Anal. Mach. Intell. **32**, 709–721 (2010)
14. Silvatti, A.P., Cerveri, P., Telles, T., Dias, F.A., Baroni, G., Barros, R.M.: Quantitative underwater 3D motion analysis using submerged video cameras: accuracy analysis and trajectory reconstruction. Comput. Methods Biomech. Biomed. Eng. **16**, 1240–1248 (2013)
15. Biswas, J., Veloso, M.: Wifi localization and navigation for autonomous indoor mobile robots. In: 2010 IEEE International Conference on Robotics and Automation (ICRA), pp. 4379–4384 (2010)
16. Faragher, R., Harle, R.: Location fingerprinting with bluetooth low energy beacons. IEEE J. Sel. Areas Commun. **33**, 2418–2428 (2015)
17. Chen, Y., Kobayashi, H.: Signal strength based indoor geolocation. In: IEEE International Conference on Communications, ICC 2002, vol. 1, pp. 436–439 (2002)
18. Amzajerdian, F., Pierrottet, D., Petway, L., Hines, G., Roback, V.: Lidar systems for precision navigation and safe landing on planetary bodies. In: Proceedings of the SPIE, vol. 8192, pp. 819202 (2011)
19. Hazas, M., Hopper, A.: Broadband ultrasonic location systems for improved indoor positioning. IEEE Trans. Mob. Comput. **5**, 536–547 (2006)
20. Rencken, W.D.: Concurrent localisation and map building for mobile robots using ultrasonic sensors. In: Proceedings of the 1993 IEEE/RSJ International Conference on IROS, vol. 3, pp. 2192–2197 (1993)
21. Titterton, D., Weston, J.L.: Strapdown Inertial Navigation Technology, vol. 17 (2004)
22. Suh, Y.S.: Attitude estimation using low cost accelerometer and gyroscope. In: Proceedings KORUS 2003 the 7th Korea-Russia International Symposium on Science and Technology, vol. 2, pp. 423–427 (2003)
23. Toyozumi, N., Takahashi, J., Lopez, G.: Trajectory reconstruction algorithm based on sensor fusion between IMU and strain gauge for stand-alone digital pen. In: 2016 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 1906–1911 (2016)
24. Wang, J.S., Hsu, Y.L., Liu, J.N.: An inertial-measurement-unit-based pen with a trajectory reconstruction algorithm and its applications. IEEE Trans. Ind. Electron. **57**, 3508–3521 (2010)

25. Park, M., Gao, Y.: Error and performance analysis of mems-based inertial sensors with a low-cost GPS receiver. Sensors **8**, 2240–2261 (2008)
26. Mourikis, A.I., Trawny, N., Roumeliotis, S.I., Johnson, A.E., Ansar, A., Matthies, L.: Vision-aided inertial navigation for spacecraft entry, descent, and landing. IEEE Trans. Robot. **25**, 264–280 (2009)
27. Lee, T., Leoky, M., McClamroch, N.H.: Geometric tracking control of a quadrotor UAV on SE (3). In: 2010 49th IEEE Conference on Decision and Control (CDC), pp. 5420–5425 (2010)
28. Bristeau, P.J., Callou, F., Vissiere, D., Petit, N.: The navigation and control technology inside the AR. Drone micro UAV. In: IFAC Proceedings, vol. 44, pp. 1477–1484 (2011)
29. Antoniou, A.: Digital Signal Processing. McGraw-Hill, New York (2016)
30. Zhang, J., Feng, J., Zhou, B.: Sensor-fusion-based trajectory reconstruction for mobile devices. In: Proceedings of the 13th International Conference on Computer Graphics Theory and Applications, pp. 48–58 (2018)