Monocular Vision-Based Autonomous Navigation System on a Toy Quadcopter in Unknown Environments

Rui Huang[1], Ping Tan[1], Ben M.Chen[2]

Abstract— **In this paper, we present an monocular vision-based autonomous navigation system for a commercial quadcoptor. The quadcoptor communicates with a ground-based laptop via wireless connection. The video stream of the front camera on the drone and the navigation data measured on-board are sent to the ground station and then processed by a vision-based SLAM system. In order to handle motion blur and frame lost in the received video, our SLAM system consists of a improved robust feature tracking scheme and a relocalisation module which achieves fast recovery from tracking failure. An Extended Kalman filter (EKF) is designed for sensor fusion. Thanks to the proposed EKF, accurate 3D positions and velocities can be estimated as well as the scaling factor of the monocular SLAM. Using a motion capture system with millimeter-level precision, we also identify the system models of the quadcoptor and design the PID controller accordingly. We demonstrate that the quadcoptor can navigate along pre-defined paths in an unknown indoor environment with our system using its front camera and onboard sensors only after some simple manual initialization procedures.**

I. INTRODUCTION

Autonomous navigation of flying robots has drawn increasing attention in recent years. Autonomous unmanned aerial vehicles (UAV) have great potential in military and civil applications such as surveillance, search and rescue, indoor mapping and 3D reconstruction. Various applications of the UAVs have been demonstrated in both indoor and ourdoor environments. Kushleyev et al. [1] achieve accurate formation control of twenty micro UAVs that have weights around 73 grams using external motion capture system. In the sFly European Project [2], a team of UAVs with downward facing cameras scan a large outdoor area and build a 3D sparse map collaboratively. Autonomous navigation of micro UAVs in GPS-denied unknown environments is still an open research question. One of the challenges is that micro UAVs have limited payload and power supply. Therefore, color cameras are often preferable sensors for the navigation control.

In terms of the autonomous navigation of UAVs using on-board sensors in GPS-denied indoor environments, L.R. Garcia Carrillo et al. [3] address the road following problem by adopting a downward looking camera for road detection and tracking. They also design switching controllers for stabilizing the pose of the UAV with respect to the road. As the camera is also used as the major sensor in their system, the approach is limited to the straight line segments appeared in the image. Our visual SLAM system is able to deal with more general environments and allows the UAV to follow arbitrary defined paths. M. Achtelik et al. [4] present a complete autonomous air vehicle system that is capable of autonomous navigation and exploration in GPS-denied environments. They design a quadrotor platform capable of carrying a laser rangefinder, stereo camera rig, color camera, and on-board computer. In contrast, we aim at a lightweight approach to autonomous navigation using only on-board sensors of a toy quadrotor. Our system can be easily integrated with micro UAVs with limited sensing payload.

Visual SLAM techniques use video cameras for simultaneously localisation and mapping. Klein et al. [5] built a system called PTAM using a color camera for visual SLAM. Newcombe et al. [6] proposed a dense tracking and mapping system which can reconstruct dense surfaces of the scene using a single camera. KinectFusion [7] is a accurate real-time SLAM system using a RGB-D camera and commodity graphics hardware. Compared to color cameras, RGB-D cameras have larger weight and higher power consumption. Despite these advances in visual SLAM, it is still challenging to use a monocular camera for autonomous navigation of an UAV. There are two major challenges. Firstly, a micro UAV has limited computation power. Thus, a ground station is often used to carry out the involved computation, with a Wi-Fi link to communicate with the drone. Since noise and signal loss during data transmission are unavoidable in real flights, the SLAM system must be robust to these problems. Secondly, the scaling ambiguity of monocular SLAM needs to be solved so that the results can be fused with the measurements from other on-board sensors.

We present an autonomous navigation system for a low cost toy quadcoptor called AR.Drone. The quadcoptor communicates with a ground-based laptop via wireless connection. The video stream of the front camera on the drone and the navigation data measured on-board are sent to the ground station and then processed by our visual SLAM system. In order to handle motion blur and frame loss in the received video, our SLAM system consists of an improved KLT tracker and a relocalisation module which achieves fast recovery from tracking failure. An Extended Kalman filter (EKF) is designed for sensor fusion to solve the scale ambiguity in visual SLAM. Our system recovers accurate 3D positions and velocities to control the drone with PID controllers. We further evaluate our system performance with a motion capture system with millimeter-level precision. It is demonstrated that the quadcoptor can navigate along pre-

[1]Rui Huang and Ping Tan are with the School of Computing Science, Simon Fraser University, Canada. {rha55, pingtan}@sfu.ca

[2]Ben M. Chen is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. bmchen@nus.edu.sg

defined paths in an unknown environment using our system.

Similar autonomous navigation system has been developed in [8] on an AR.Drone. In their system, an extended Kalman filter (EKF) is designed to fuse all available measurements. In order to compensate the data transmission latency, drone pose is predicted by a heuristic motion model assuming fixed latency. However, in practice, we found the data transmission latency varies over time. Furthermore, [8] uses a standalone scale estimator to solve the scaling ambiguity. Specifically, given a time interval, both of the visual SLAM and on-board metric sensors estimate the distance traveled by the drone. Assuming both of the measurements contain Gaussian noise with constant variance, the scale can be solved by a maximum likelihood approach. This approach to solve the scale ambiguity relies heavily on the drone's vertical motion, because it requires accurate altitudes from the ultrasonic sensor. However, vertical motion is not common during indoor navigation. Nützi et al. [9] present a different method to solve the scale ambiguity of visual SLAM. They also designed an EKF, where the scale presents in the state space and is estimated continuously. The scale is computed by comparing the position from visual SLAM and the position from integrating IMU measurements. This scale estimation suffers from drifting errors caused by integration. Motivated by both of the approaches, we design a novel EKF to fuse the visual SLAM and velocity measurement from the on-board sensors. The position estimated by visual SLAM is able to correct the position drifting caused by integrating the velocity measurements. The velocity measurements in turn provide good reference for the scale estimation. Our EKF fuses both measurements to solve the scale ambiguity and minimize drifting errors. At the same time, we design our visual SLAM algorithm to make it robust to data communication errors and latencies.

The remaining of the paper is organized as following. The platform and software used along with our system is discussed in Section 2. Our visual SLAM system is running on a ground-based laptop. We adopts the idea from PTAM to separate the mapping and tracking process for better efficiency. To achieve better robustness to data transmission problems, we enhance the feature tracking and include a re-localization component. The details of our visual SLAM system are discussed in Section 3. As described by Bristeau et al. [10], the AR.Drone estimates the linear velocities in the horizontal plane of the body frame with its on-board sensors. These velocities are estimated by a navigation filter integrating the measurements of a downward looking camera and inertial sensors. We take these on-board measurements and fuse them with the results of our visual SLAM system using a novel Extended Kalman filter which is elaborated in Section 4. Section 5 presents experiment evaluations. The videos demonstrating the autonomous path following are available online:`http://www.sfu.ca/~rha55/`.

II. PLATFORM

The AR.Drone is a micro Unmanned Aerial Vehicle (UAV) developed by the Parrot company for the increasing market of videos games and home entertainment. The drone is low-cost, safe, and easy to fly by end users. These properties also make it a suitable platform for our research.

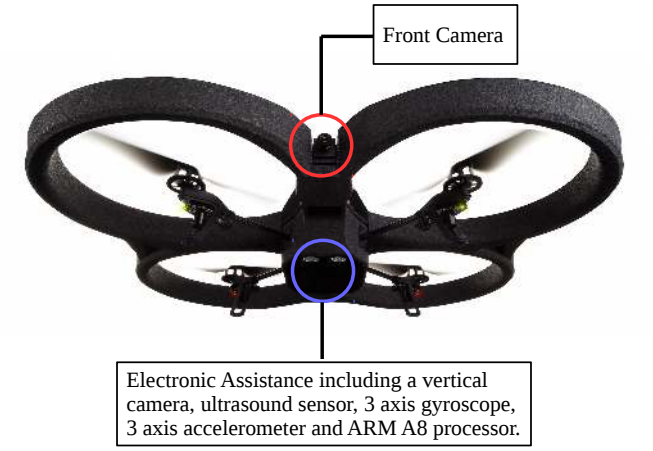


Fig. 1. The Parrot AR.Drone 2.0.

A. Hardware

The AR.Drone platform is detailed in [10]. We briefly summarize it here to make this work self-contained. The drone is based on a classic quadrotor design with four brushless motors. The on-board electronics consists of two boards, i.e., Mother-board and Navigation board. The mother-board is equipped with an ARM9-core processor running at 468 MHz. A Linux based operating system and all on-board computations are run on the processor. The mother-board is connected to two cameras including a front camera and a downward-looking camera. The front camera has a view angle of 93 degrees and a VGA resolution of 640×480. It can capture at up to 30 Hz. The vertical camera is a 64 degrees diagonal lens camera used to perform the estimation of the vehicle speed. The navigation board is equipped with the sensors including a 3-axis accelerometers, a 2-aixs gyroscope, a 1-axis vertical gyroscope and 2 ultrosonic sensors. The ultrasonic sensors provide the altitude estimation and the vertical displacements of the UAV. The embedded operating system manages the Wi-Fi communications between the UAV and the ground station. It is also in charge of video sampling and compression, image processing, sensors acquisition, state estimation and closed-loop control. Once a computer is connected to the broadcasting Wi-Fi network, the Wi-Fi chip can transmit sensor data and estimated states at 200Hz, and compressed video frames at up to 30Hz. The important components of the AR.Drone is highlighted in Figure 1.

B. Platform Modeling and Controller Design

The attitude stabilization of the drone is performed on-board and realized by the attitude control loop and the angular rate control loop [10]. The attitude reference is set by sending floating number parameters between $[-1, 1]$ to the four channels including roll angle, pitch angle, vertical speed, and yaw rate.

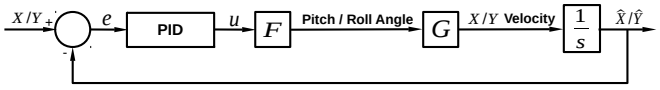


Fig. 2. Control loop for X and Y positions.

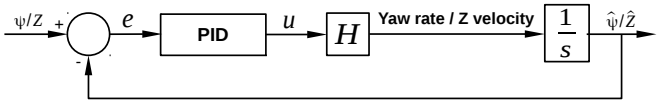


Fig. 3. Control loop for yaw angle ψ and altitude Z.

In order to design the outer loop controllers, we need to perform system modeling for the on-board controllers. The control loop for X, Y positions is illustrated in Figure 2. The control loop for vertical altitude and yaw angle is shown in Figure 3. We use the System Identification Toolbox of MATLAB to fit the system transfer functions for each channel separately. Firstly, we sample a sequence of a chirp signal as the control parameters. Then we send the control signals at the sampling rate to the drone and record the response using a motion capture system called Vicon with millimeter precision. The input control signals of roll angle and corresponding roll measurements are plotted in Figure 4. The input and output signals are then processed by the System Identification Toolbox to generate the transfer functions.

In Figure 2, the PID controller receives position error e in the local North-East-Down (NED) frame of the drone and computes control signal u. The transfer function F describes the relation of control signal u and roll or pitch angles. Then the velocity along X or $Y - axis$ can be predicted by G. By integrating the velocity, the predicted position can be obtained. Figure 3 shows the control loop applied to yaw angle and altitude control. The control signal u encodes the desired yaw rate or vertical velocity. Thus, the transfer function H estimates the yaw rate and vertical velocity based on the control signals. The identified transfer functions are

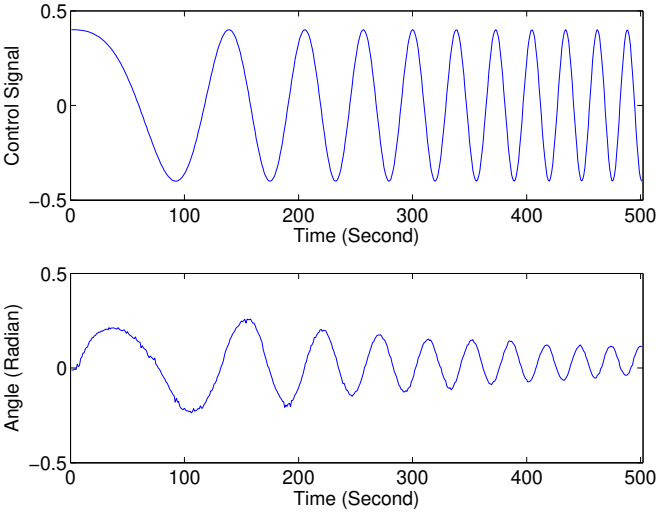


Fig. 4. Input control parameters and roll angle measurements by Vicon.

listed as following.

$$F_{\mathbf{x}} = \frac{0.76034}{0.0247s^2 + 0.3144s + 1}, \quad G_{\mathbf{x}} = \frac{-15.945}{1.6583s + 1} \tag{1}$$

$$F_{\mathbf{y}} = \frac{0.74113}{0.0215s^2 + 0.2935s + 1}, \quad G_{\mathbf{y}} = \frac{10.130}{1.0569s + 1} \tag{2}$$

$$H_{\mathbf{alt}} = \frac{0.879}{0.0619s^2 + 0.2608s + 1} \tag{3}$$

$$H_{\mathbf{yaw}} = \frac{1.5203}{0.0359s + 1} \tag{4}$$

With the identified system models, we solve our PID controllers with the filter coefficient parameter N using the Simulink software of MATLAB. The discrete PID controller in our implementation is presented below. The control signal u_t is generated at 25 Hz based on the position or angle error e_t. the sampling time $\Delta t = \frac{1}{25}$ seconds. The reference state $\mathbf{x}_{\text{ref}}$ consists of the drone's 3D position and its yaw angle.

$$u_t = P_t + I_t + D_t \tag{5}$$

$$P_t = e_t \cdot K_{\text{p}} \tag{6}$$

$$I_t = \int e_t \Delta t \cdot K_{\text{i}} \tag{7}$$

$$D_t = \frac{D_{t-1} + K_{\text{d}} \cdot N \cdot (e_t - e_{t-1})}{1 + N \cdot \Delta t} \tag{8}$$

$$e_t = \mathbf{x}_{\text{ref}} - \mathbf{x}_t \tag{9}$$

The gains of the PID controllers used in our implementation are presented in Table I. Note that the control gains of X Position Controller are negative. With the driver we are using to control the ARDrone, positive control signals of the pitch angle make the drone move towards the negative $X - axis$. So the error in X position and the control signal have opposite signs.

TABLE I

PARAMETERS OF OUR PID CONTROLLERS

Controller	K_{p}	K_{d}	K_{i}	N
X Position Controller	-0.144	-0.267	-6.362e-6	203.645
Y Position Controller	0.247	0.241	1.595e-5	186.796
Yaw Angle Controller	6.564	0	0	0
Altitude Controller	0.600	0.100	0.001	138.861

III. VISUAL SLAM

Our visual SLAM system consists of a highly efficient and robust feature tracking component. The feature tracks are then used to register the current camera position to a global 3D map. From time to time, new 3D map points are triangulated and new key-frames are inserted, to enhance system robustness. A local Bundle Adjustment (BA) optimize the global map in a parallel thread. Our SLAM system also includes a re-localisation module, which enables it to recover from tracking failure.

A. Feature Tracking

Our feature tracker detects strong corners [11] and track them by a sparse iterative version of the Lucas-Kanade optical flow [12]. To facilitate exploration of new environments, it detects new corners every five frames. A feature track will be terminated whenever its correspondence cannot be found. The drone's motion could cause blurry frames, especially in relatively dark indoor environments where longer exposure is required. Blurry frames leads to rapid and significant drop in the number of tracked corners, and hence poor camera pose estimation. A re-localization could address this problem, while it is relatively computationaly expensive. We observe most of the blurs last only a few frames (≤ 3), so we design a 'rollback' scheme for the tracker to deal with frequent and short blurs. If the number of tracked corners drops significantly at the k-th frame, we try to track from the frames in $[k-5, k-1]$ to the frames in $[k+1, k+3]$. If it succeed, re-localization will not be called.

This rollback scheme cannot solve all tracking problems. Sometimes, multiple continuous video frames could be lost due to poor Wi-Fi connection. When this happens, none of the previously visited frames can be tracked to the current frame. Such an example is provided in Figure 7, where the two received consecutive frames have quite a large motion. In this case, our system will activate the re-localization module to recover from the tracking failure. Specifically, we count the number of feature tracks with 3D information in the current frame. If the percentage of these feature tracks drops below a threshold, the re-localisation will be triggered. The details of our re-localisation method is in section III-D.

B. Camera Pose Estimation

The camera intrinsics are pre-calibrated and fixed. We solve the camera extrinsics $\mathbf{\Theta} = (\mathbf{R}, \mathbf{t})$ by minimizing the following reprojection error,

$$\mathbf{\Theta}_{\text{opt}} = \arg\min_{\mathbf{\Theta}} \sum_i \rho(||\mathbf{m}_i - \mathrm{P}(\mathbf{M}_\mathrm{i}, \mathbf{\Theta})||), \qquad (10)$$

where $\mathrm{P}(\mathbf{M}_\mathrm{i}, \Theta)$ projects a 3D map point $\mathbf{M}$ to the image plane and $\mathbf{m}$ is its corresponding image feature. The function ρ is the Tukey bi-weight function [13]. We minimize Equation 10 by the iteratively re-weighted least squares (IRLS) method. The Θ is initialized as the camera pose at the previous frame. At each iteration of the IRLS, the Levenberg-Marquart algorithm is used to solve the non-linear least square problem.

C. Keyframe Insertion and Map Generation

The current frame will be inserted as a keyframe if it satisfies the following criteria. Firstly, it is separated from the last keyframe for a certain interval. Secondly, the number of tracked 3D map points in it is below a threshold. Furthermore, the number of unmapped feature tracks are more than a threshold. We cache a keyframe as the full size image, a downsized thumbnail image, and its tracked corners and their corresponding 3D map points.

New map points will be generated when the tracking is stable while the number of tracked map points is dropping. We evaluate the stability of tracking by the ratio of the number of tracked corners in the current frames and that number in the previous frame. If this ratio is higher than a threshold, we consider the tracking is stable. New map points are triangulated [14] using the feature points at the head and tail of feature tracks. The triangulated map points are accepted only if their reprojection errors are smaller than a threshold, they are not behind of any cameras, and the triangulation angle between the two cameras is larger than a threshold.

D. Relocalisation

Re-localization is critical for the system robustness. During re-localisation, each available input frame will be downsized and compared with the cached thumbnails of the keyframes. We set the width of the thumbnails to be 100 and the threshold of the sum of squared difference (SSD) between two thumbnails to be 100. If a similar keyframe K is successfully found for the input frame F, the current camera position will then be registered to the global map via the keyframe K. In this way, the SLAM system can recover from tracking failures.

To register K and F, we search feature matching between them. In order to obtain high quality matching within reasonable processing time, we firstly detect FAST features [15] in F and K. We then compute the SIFT descriptors [16] at these features and match them within a local neighborhood with diameter of 6 pixels. Then the two sets of descriptors are matched using the FLANN matcher [17]. Although the algorithm is designed to produce high quality feature matching, there are still outliers in the output feature matches. We further eliminate the outliers by fitting the essential matrix with RANSAC alorithm [14].

In order to boost the number of feature matchings, we detect additional correspondences by masking the existing ones. These new features and their corresponds will be triangulated to generate additional 3D map points. The triangulated 3D point is valid if its reprojection errors are smaller than 3 pixels. Otherwise, it is discarded and the corresponding feature match is considered as an outlier. Empirically, we find this step is important, since it ensures sufficient features with 3D information when the system is recovered. Without it, the system is likely fail again soon and re-localization will be triggered repetitively. The pipeline of our vision system is illustrated in Figure 5

IV. SENSOR FUSION

The results of visual SLAM are up to a scale. This scaling ambiguity must be solved so that the reconstructed 3D information can be used for navigation control. On the other hand, the on-board sensors of an AR.Drone estimate the drone's altitude and its velocities on the horizontal plane. In principal, we could obtain the drone's position by

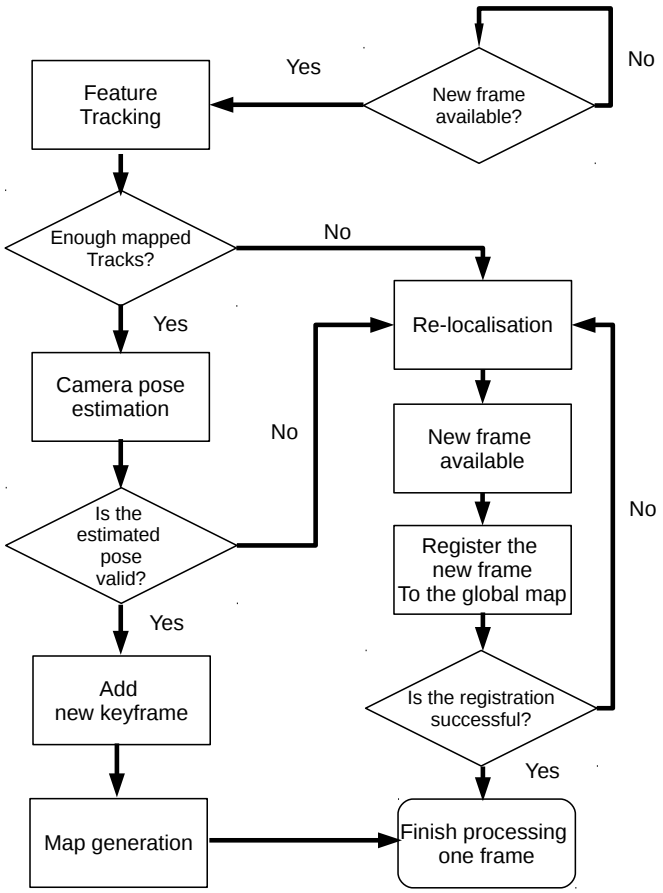


Fig. 5. SLAM system pipeline

integrating its horizontal velocities over time. However, such an integration suffers from bad drifting errors. We describe a sensor fusion algorithm based on extended Kalman filter to fuse the results from visual SLAM and the outputs from onboard sensors.

While Engel et al.[8] also employed EKF for sensor fusion and scale estimation, our solution has the following advantages. Firstly, we use a much simplified state space to reduce the computation, thanks for the good accuracy of the on-board attitude and velocity measures. Secondly, unlike [8], our scale estimator does not rely much on vertical motion of the drone, which is not common in the real flights. Our proposed EFK is inspired by Nützi et al. [9], where the scale is estimated by comparing the results of visual SLAM and integration of IMU measurements. To address the drifting errors in integrating IMU measurements, we solve the scale factor by comparing velocities from visual SLAM and onboard sensors.

A. Our EKF

In this section, we introduce the state space of the EKF followed by the prediction model and the measurement model. We empirically found the attitude returned from the drone's onboard sensor is quite robust. So the state space only consists of the position $\mathbf{p}$ in meter, velocity $\dot{\mathbf{p}}$ in meter per second and scaling factor s in meter. These are the only inputs to our controller for generating control commands. The state space is:

$$\mathbf{x} = \begin{bmatrix} p \\ \dot{p} \\ s \end{bmatrix}. \tag{11}$$

This state space involves the one-dimensional position p and velocity $\dot{p}$ for the sake of simplicity. However, it is easy to be extended to multiple dimensions.

The prediction model is:

$$\mathbf{x}_{k|k-1} = f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{w}_{k-1} \tag{12}$$

There is no $\mathbf{u}$, since we do not have an input control. This prediction model can be further expressed as the following,

$$\mathbf{x}_{k|k-1} = \begin{bmatrix} 1 & \Delta t & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{x}_{k-1} + \mathbf{G} \cdot \mathbf{a}_{k-1} \tag{13}$$

and

$$\mathbf{G} = \begin{bmatrix} \Delta t^2/2 \\ \Delta t \\ 0 \end{bmatrix} \tag{14}$$

In this constant acceleration model, we assume the acceleration $\mathbf{a}$ of the drone is constant during Δt and follows a zero-mean normal distribution with standard deviation. Therefore, the covariance $\mathbf{Q}$ of the state transition noise $\mathbf{w}$ can be computed as the following,

$$\mathbf{Q} = \mathbf{G}\mathbf{G}^T {\sigma_a}^2 \tag{15}$$

The state transition matrix is defined by,

$$\mathbf{F} = \begin{bmatrix} 1 & \Delta t & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{16}$$

and the predicted estimate covariance is obtained by,

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} {\mathbf{F}_k}^T + \mathbf{Q}_k. \tag{17}$$

The measurement model considers two measurement sources, i.e., visual tracking and onboard sensors. We obtain the estimated position directly from the visual SLAM system. To compute the velocity at time t measured by the visual SLAM system, we differentiate the positions $\mathbf{p_t}$ and $\mathbf{p_{t'}}$ at frames N_t and $N_{t'}$. The time stamps t and t' are recorded with the video frames. The frame difference ΔN is set to 30 in our experiments to avoid sudden jump in velocity estimation.

The first measurement model for the position p_v and velocity $\dot{p}_v$ from the visual SLAM system is,

$$\mathbf{z}_v = h(\mathbf{x}_v) + \mathbf{v}_v \tag{18}$$

where

$$h(\mathbf{x}_v) = \begin{bmatrix} 1/s & 0 & 0 \\ 0 & 1/s & 0 \end{bmatrix} \cdot \mathbf{x}_v, \tag{19}$$

$$\mathbf{z}_v = \begin{bmatrix} p_v \\ \dot{p}_v \end{bmatrix}, \tag{20}$$

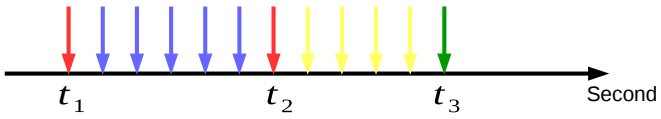


Fig. 6. Measurement update scheme

and $\mathbf{v}_v$ is the observation noise of the visual tracking. Thus the observation matrix is

$$\mathbf{H}_v = \begin{bmatrix} 1/s & 0 & -s^{-2} \cdot p \\ 0 & 1/s & -s^{-2} \cdot \dot{p} \end{bmatrix} \tag{21}$$

The second measurement model for the estimated velocity $\dot{p}_n$ from the navigation data is,

$$\mathbf{z}_n = h(\mathbf{x}_n) + \mathbf{v}_n \tag{22}$$

where

$$h(\mathbf{x}_n) = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \cdot \mathbf{x}_n, \tag{23}$$

$$\mathbf{z}_n = \dot{p}_n, \tag{24}$$

and $\mathbf{v}_n$ is the observation noise of the navigation data obtained from the onboard sensors. Thus the observation matrix is

$$\mathbf{H}_n = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}. \tag{25}$$

Once there is a new measurement from either visual tracking or onboard sensors, the EKF will be updated. The innovation residual and covariance are,

$$\mathbf{y}_k = \mathbf{z}_k - h(\mathbf{x}_{k|k-1}) \tag{26}$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} {\mathbf{H}_k}^T + \mathbf{R}_k \tag{27}$$

The Kalman gain is computed as,

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} {\mathbf{H}_k}^T {\mathbf{S}_k}^{-1} \tag{28}$$

Finally, the state x and covariance P are updated by,

$$\mathbf{x}_k = \mathbf{x}_{k|k-1} + \mathbf{K}_k \mathbf{y}_k \tag{29}$$

and

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \tag{30}$$

respectively.

B. Filter Update Scheme

We now consider the update scheme of different measurements. The AR.Drone transmits the navigation data and video frames to the ground station for processing. Both data transmission and processing cause latency. We then propose a measurement update scheme for compensating these delays.

The navigation data including the estimated velocity are buffered in a queue once received. The EKF will only be updated permanently when a measurement from the visual tracking is ready. The updates are based on the time stamp when the data are received from the drone. Then we use the buffered navigation data to predict the states to the current time.

As illustrated in Figure 6, red arrows represent the updates of the EKF using available measurements from visual tracking at t_1, t_2. The blue arrows in between represent the updates of the EKF using the buffered measurements from navigation data. To obtain a better estimation of the current states before the next visual measurement is available, we only use the buffered navigation data received after t_2 to do the prediction as shown by yellow arrows. The state of the EKF will be updated up to time t_3 which is closest to the current time. The latest measurement used for update is shown in the green arrow.

Fig. 7. The video is discontinuous at frame 508 and 509 due to the frame loss caused by unstable Wi-Fi link

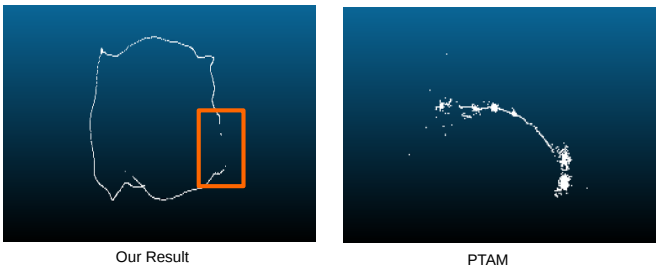


Fig. 8. The left figure shows our results of camera position reconstruction. The relocalisation algorithm is activated in the period highlighted by the orange rectangle. The right figures shows the failed result of PTAM.

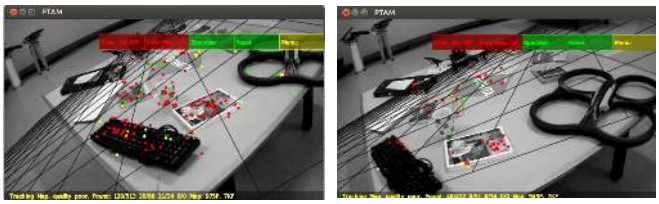
V. Experiments

Our visual SLAM system can achieve real-time performance and the control signal is generated at 25 Hz on the ground station and sent to the drone. We first evaluate the individual SLAM and EKF components respectively. Then, the whole system is tested in real flights and the ground truth positions measured by the Vicon system are examined. The experiment environment is shown in Figure 14.

A. SLAM system

Our system is robust to feature tracking failures which can happen due to multiple reasons, including blurry frames and frame loss in transmission. A typical example of frame loss is shown in Figure 7, where two received consecutive frames have significant change to make feature tracking fail. Our system is robust to these kinds of problems, owing to the 'rollback' scheme in tracking and the re-localization module.

We show a comparison with the well known monocular SLAM system PTAM [5] on the videos recorded by our AR.Drone. Figure 8 shows the reconstructed camera trajectories from both systems. In this example, the drone is facing a table and moves surround it and then back to the starting point. Our system can successfully reconstruct the camera trajectory for the whole sequence. Note that the position



Frame 508 Frame 509

Fig. 9. Snapshots of tracking from PTAM

Frame 508 Frame 509 Frame 517

Fig. 10. Snapshots of tracking from our system

jumps as highlighted by the orange rectangle in the left of Figure 8. This sudden change in position is due to the frame loss. Our method can deal with this type of problems better.

Empirically, we find our feature tracking is superior than that in PTAM. PTAM's feature tracker adopts a motion model to predict the feature's location in the coming video frame. It only search correspondences nearby the predicted feature location. In practice, we find this pre-defined motion model is often inconsistent with the drone's motion. Thus, the true feature is significantly different in orientation and scale with the predicted one. This problem makes PTAM's feature tracking fragile. As presented in Figure 9, the sudden view change causes a significant drop in the number of tracked features for PTAM. The estimated plane and the pose are obviously wrong. In comparison, our method is free from this problem. As shown in Figure 10, we indicate the tracked feature points with corresponding 3D map points in green. A sudden viewpoint change will cause significant drop in the number of green points (see the middle of Figure 10). Once that happens, our relocalisation will be triggered to revive the system (see the right of Figure 10).

Figure 11 shows another challenging example, where the camera makes a quick turn at a corner. As the scene changes quickly during the navigation, it is difficult for a visual SLAM system to register map points to image features. Our system tracks feature well and detect new features whenever the number of tracked feature drops significantly. Furthermore, our map points triangulation is not limited to the keyframes. New map points can be generated timely and added to the map. This makes the system robust to scene occlusion and large view change. Figure 12 shows the comparison of the results generated by our system and PTAM. It is clear that our system also tracked more feature points than PTAM as shown in Figure 13.

B. EKF based Sensor Fusion and Scale Estimation

We evaluate our proposed extended Kalman filter in this section. Our EKF directly provides the position estimation of the drone in real metric unit. The scaling ambiguity of the visual SLAM results is solved as a state in the filter. We

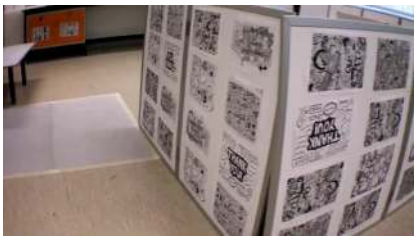
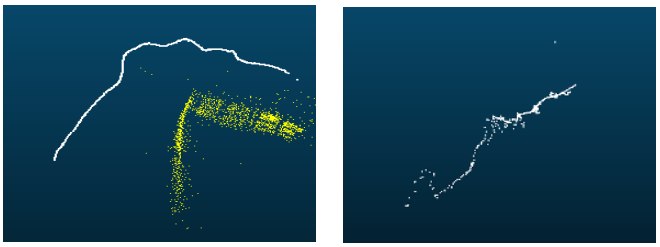


Fig. 11. Camera moving around a corner formed by two planes with textures

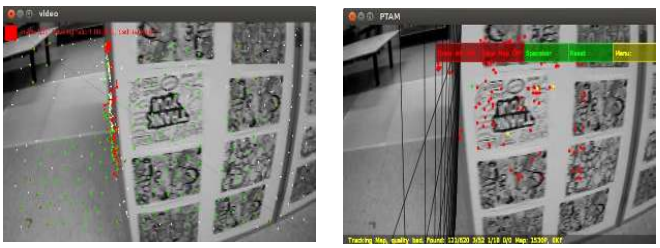


Our Result PTAM

Fig. 12. The left figure shows our results of camera position reconstruction. The right figures shows the result of PTAM. PTAM failed when the camera is turning at the corner

compare the estimated positions with Vicon measurements to have quantitative evaluation for our approach. This result is shown in Figure 15. In this flight, the drone is mainly navigating along the $Y - axis$. We register the Vicon coordinate system to the one used in our visual SLAM, so that the Vicon measurements can be transformed to the North-East-Down (NED) frame of the drone. The blue curves shows the Vicon measurements of position in $Y - axis$ and our estimated positions are in red. Our estimated positions match the Vicon measurement closely. For a comparison, we also plot the positions computed by directly integrating the onboard velocity measurement. It is clearly shown that the integration suffers from large drifting errors.

Figure 16 shows the state space of the EKF along $Y - axis$. The third row presents the estimation of scaling factor. Although the initial state is far from the correct scale, the filter can converge in less than 15 seconds. The initialization process can be done manually before the autonomous navigation starts.



Our Result PTAM

Fig. 13. During the navigation, our system (Left) tracked more features than PTAM (Right).

Fig. 14. The experiments are performed in an unknown environment. The textures pasted in the environment are only used to boost the features in the original environment with poor textures.

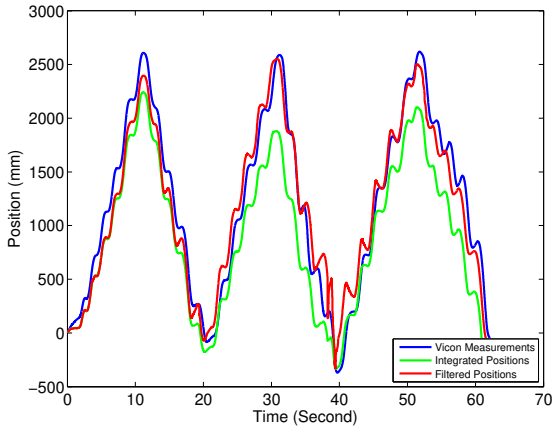


Fig. 15. The blue curves shows the Vicon measurements of position in $Y-axis$ and our estimated positions are in red. The integrated position in green shows a large drifting error.

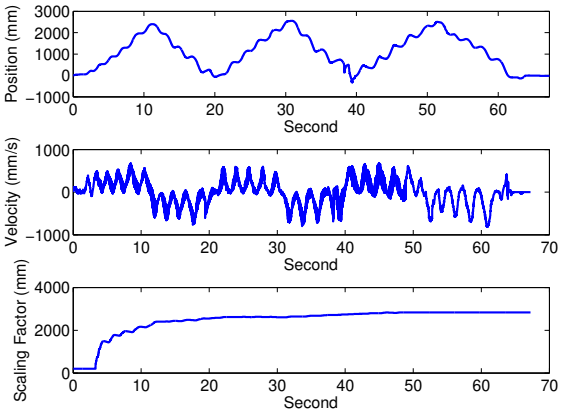


Fig. 16. The state space of the EKF

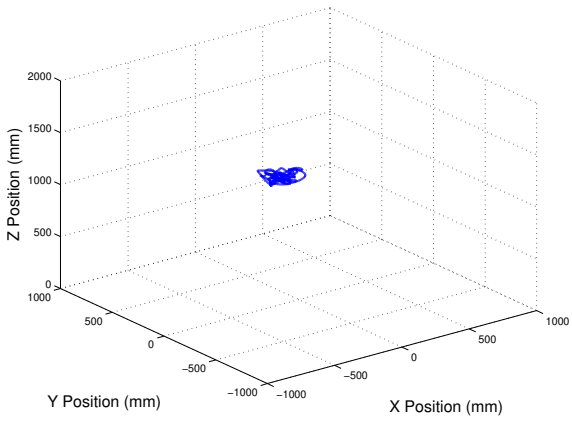


Fig. 17. 3D position of the UAV measured by Vicon system in the hovering test

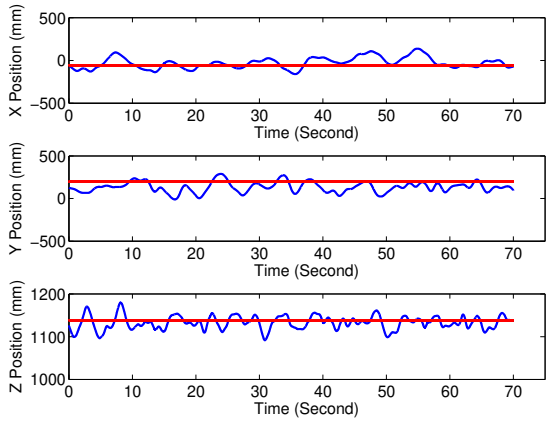


Fig. 18. 2D plot of the hovering test result. The UAV position measured by the Vicon system is shown in blue and the set point is denoted by the red line.

C. Complete System Performance

We firstly verify our system with a hovering test in an unknown indoor environment. After initializing the visual SLAM system manually, we sent the hovering command to the AR.Drone. The drone then tries to hold to the reference position using the estimated pose. Meanwhile, we use the Vicon system to record the precise positions of the flying drone. The root-mean-square-error (RMSE) of the drone's position is computed for quantitative evaluation. In this hovering test, the RMSE of the position errors are $70.1821mm$ in $X-axis$, $86.51mm$ in $Y-axis$, and $16.75mm$ in $Z-axis$. The error is smaller in the $Z-axis$ because our EKF module fuses the altitude measure of the ultrasonic sensor. Figure 17 shows the position of the UAV in 3D space. Figure 18 demonstrates the positions measured by Vicon system (blue) and the reference positions (red) along individual axis.

We further carry out path following experiments to make the drone autonomously follow a pre-defined 3D path. The input path is generated with a sequence of set-points with reference positions and velocities. We adopted a similar approach used in [18] and [19] to generate the reference trajectory with continuous and limited velocity and acceleration.

RMSE	X-axis	Y-axis	Z-axis
Square: path tracking (mm)	178.578	122.28	16.84
Square: position estimation (mm)	190.20	139.40	28.40
Circle: path tracking (mm)	81.31	98.91	18.88
Circle: position estimation (mm)	190.63	227.79	72.63

TABLE II
PERFORMANCE EVALUATION

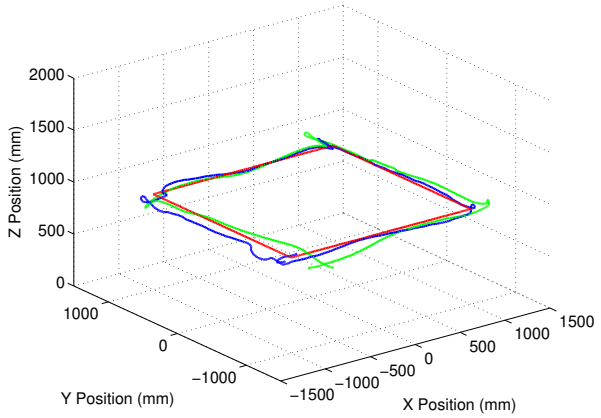


Fig. 19. 3D position of the UAV measured by Vicon system in the square path following test. The path reference is shown in red. The Vicon measurements are in green. The estimated positions are in blue.

The generated reference path includes continuous reference positions, velocities and accelerations. Given user-specified control points, we optimize the overall time span of the trajectory to obtain a time optimal path. The B-spline based optimization algorithm is adopted to constrain the derivatives of the continuous trajectory. The optimization problem is solved by non-linear programming. The generated path prevents spikes in control input and helps the localization algorithm to achieve a stable and smooth performance. We sample the reference points on the generated path with the control frequency and update the reference point with the same frequency. The PID controller generates control signal according to the current reference point. The experiments show that the UAV is able to follow the reference trajectory smoothly using our navigation system.

We tested two flight paths including a $2m \times 2m$ square and a circle with the radius of $1m$. Figure 19 and 20 present the performance of the square path following. The circle path following results are shown in Figure 21. We record the Vicon measurements of the drone's position for quantitative evaluation. We evaluate both the path tracking performance and the position estimation accuracy in Table II. It is shown in the experiments that our system can produce reasonably accurate position estimation for indoor navigation of the UAV.

VI. CONCLUSION

We have presented an autonomous navigation system for a low-cost toy UAV, the Parrot AR.Drone. This paper gives detailed descriptions of the platform and system modeling using a motion capture system. Our visual SLAM system

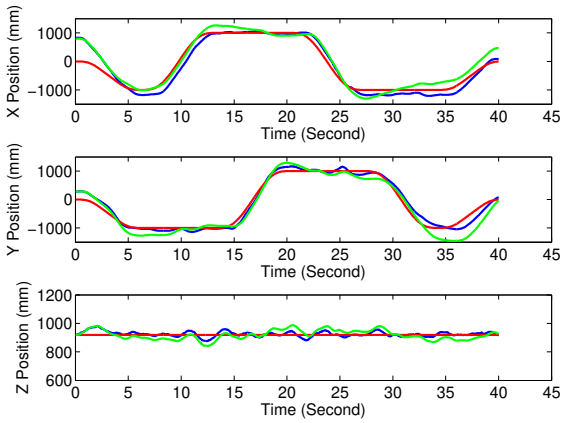


Fig. 20. The result of square path following test. The position reference is shown in red. The Vicon measurements are in green. The estimated positions are in blue.

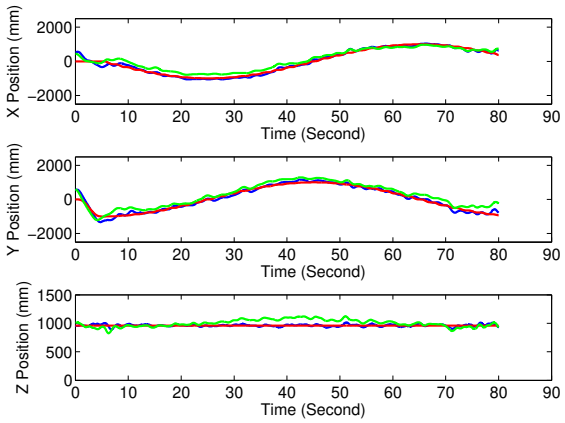


Fig. 21. The result of circle path following test. The position reference is shown in red. The Vicon measurements are in green. The estimated positions are in blue.

contains a robust feature tracker and a relocalization module to estimate accurate poses in real time. The proposed EFK fuses measurements from the visual SLAM and the on-board navigation filter to correct the local drifting error and solve the scaling ambiguity. It has been shown in the experiments that the AR.Drone can perform robust autonomous navigation using our system. The results have been verified in real flights with ground truth positions measured by Vicon. The position accuracy is within 10 cm, which is reasonable for autonomous indoor navigation of UAVs. Our system can be easily adapted to other platforms with inertial sensors and video transmitter.

REFERENCES

[1] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors." *Autonomous Robots*, pp. 287–300, 2013.

[2] M. Achtelik, Y. Brunet, M. Chli, S. Chatzichristofis, J.-d. Decotignie, K.-m. Doth, F. Fraundorfer, L. Kneip, D. Gurdan, L. Heng, *et al.*, "Sfly: Swarm of micro flying robots," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 2649–2650.

[3] L. R. G. Carrillo, G. Flores Colunga, G. Sanahuja, and R. Lozano, "Quad rotorcraft switching control: An application for the task of

path following," *Control Systems Technology, IEEE Transactions on*, vol. 22, no. 4, pp. 1255–1267, 2014.

[4] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Autonomous navigation and exploration of a quadrotor helicopter in gps-denied indoor environments," in *First Symposium on Indoor Flight*, no. 2009, 2009.

[5] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE, 2007, pp. 225–234.

[6] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2320–2327.

[7] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE, 2011, pp. 127–136.

[8] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 2815–2821.

[9] G. Nützi, S. Weiss, D. Scaramuzza, and R. Siegwart, "Fusion of imu and vision for absolute scale estimation in monocular slam," *Journal of intelligent & robotic systems*, vol. 61, no. 1-4, pp. 287–299, 2011.

[10] P.-J. Bristeau, F. Callou, D. Vissière, N. Petit, *et al.*, "The navigation and control technology inside the ar. drone micro uav," in *18th IFAC world congress*, vol. 18, no. 1, 2011, pp. 1477–1484.

[11] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994, pp. 593–600.

[12] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 5, 2001.

[13] Z. Zhang, "Parameter estimation techniques: A tutorial with application to conic fitting," *Image and vision Computing*, vol. 15, no. 1, pp. 59–76, 1997.

[14] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[15] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision–ECCV 2006*. Springer, 2006, pp. 430–443.

[16] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[17] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration." in *VISAPP (1)*, 2009, pp. 331–340.

[18] K. Li, R. Huang, S. K. Phang, S. Lai, F. Wang, P. Tan, B. M. Chen, and T. H. Lee, "Off-board visual odometry and control of an ultralight quadrotor mav," in *IMAV 2014: International Micro Air Vehicle Conference and Competition 2014, Delft, The Netherlands, August 12-15, 2014*. Delft University of Technology, 2014.

[19] S. K. Phang, S. Lai, F. Wang, M. Lan, and B. M. Chen, "Uav calligraphy," in *Control & Automation (ICCA), 11th IEEE International Conference on*. IEEE, 2014, pp. 422–428.