State Estimation for Swarm UAVs Under Data Dropout Condition

Hongzhe Yu, Weifan Zhang, Xinjun Sheng(✉), and Wei Dong

Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai, China
{ben0107,zwf_jaccount,xjsheng,dr.dongwei}@sjtu.edu.cn

Abstract. In this work, a method based on position predicting, velocity filtering and self adaptive parameter tunning is addressed for state estimation and control for swarm of mini unmanned aerial vehicles (UAVs), in order to deal with random noise and data dropout appeared during flights. Under conditions of random data dropout rates and communication latencies, the presented algorithm gives position prediction based on filtered velocity estimation and it fuses the prediction with sensor data. At the same time it corrects the prediction by the error between prediction and measurement of the previous step. The algorithm is designed for tracking mini UAVs with identical marker configuration, and the principles refered is in potential of serving to state estimation in various circumstances. Based on this localization algorithm, a cascade nonlinear control model is developed for swarm UAV control. This work contributes mainly to the object localization and control in a multi-agent system in which all the agents are considered to be in an identical form, hoping that this work will be the testbed for more complicated swarm robot control experiments. Comparison results of state estimation are presented by implementing experiments with or without data dropout.

Keywords: State estimation · Data dropout compensation
UAV control

1 Introduction

1.1 Swarm UAV

In the recent years, unmanned aerial vehicles (UAVs) swarm has been attracting lots of attention. The deployment of an UAVs swarm is able to accomplish much more complex and difficult tasks than a single vehicle. Considerable promising applications are expanded in both military and civil areas. For example, UAV swarms can operate within military missions under a mission-based framework [1]. UAV swarms can also be deployed for patrolling and monitoring an area of people [2] or an area of wild-life [3]. As for post-disaster management,

H. Yu and W. Zhang—Equal contributors.

Z. Chen et al. (Eds.): ICIRA 2018, LNAI 10984, pp. 81–91, 2018.
https://doi.org/10.1007/978-3-319-97586-3_7

UAVs can play a crucial role in disaster response by mapping terrain, estimating damage, etc. [4] Several applications in agriculture domain are also presented recently. Loayza *et al.* performs a sowing seeding task using a centralized UAVs swarm [5], and a swarm of collaborating UAVs are designed for field coverage and weed mapping by Albani *et al.* [6]. In general, UAVs swarm includes various technologies and theories, such as robot sensing, data fusing, cooperative optimization, information network and so on. Especially with the development of drone markets, UAVs swarm are highly regarded by different research areas.

Over the last few years, many multi-robot systems have been presented, which can be classified into two typical types according to the global information accessibility [7]. One is the robotic swarms whose agents only have access to local information and limited communication ability. The other type is called general multi-robot system and its agents are able to obtain global information and all-to-all communication. Another sorting principle is based on the communication structure. Under this principle, multi-robot systems are divided into three subclasses, i.e., centralized systems, decentralized systems, and semi-centralized system (semi-centralized systems are also known as leader-follower structure [8]). Some quick conclusions can be drawn from two classifications above. 1. All of the robotic swarms are decentralized. It requires every robot to have at least one active position sensor so that they can build up the system. 2. A general multi-robot system can either be centralized or decentralized. Since decentralized algorithms can be transformed into a variety of centralized algorithms by employing a systematic methodology in the all-to-all communication condition [9], both active and passive position sensors are feasible for general multi-robot systems.

In this paper, we mainly discuss the centralized general multi-robot system. One major problem of this type of systems is data dropout and many compensation solutions have been put forward.

1.2 Related Work

For indoor state estimation, different methods and solutions have been proposed. A decentralized localization system using ultra-wideband radio triangulation [11] is introduced but with error in position (about 10 cm) that is not tolerable by swarm formation flights. James A. Preiss *et al.* introduce an Iterative closest point (ICP) frame-to-frame tracking method to track uavs [12], which provides a method to register a newly obtained point cloud to a previous one. This method considers the compensation for temporary communication latencies by implementing an Extended Kalman filter (EKF) to fuse data from motion capture system and IMU measurement, which requires the completeness of information in each frame obtained (with no point dropout) during flying. Though similar in key components, the work mentioned in this paper differs from previous works in the way that it utilizes the geometry characteristic of the markers and fuses the estimated velocity and acceleration to serve for prediction of the state estimation in the next frame by which it can compensate for possible noise and data dropouts. Random marker dropout (easily caused by overshadowing among

agents in tightened formations) and other noisy conditions (random occurrence of reflections which are visible to the motion-capture cameras, etc.) reduces the robustness of the state estimations if not taken into considerations.

2 State Estimation

Compared to outdoor circumstances, indoor motion capture systems such as vicon provides much higher precision in position estimation [10], which allows the realization of complicated and precise control algorithms. However, typical tracking systems provides only the precisely measured markers. In presence of a multiple objects, most tracking systems fail to provide robust single-point object tracking algorithms. Furthermore, when facing with multiple objects, main existing tracking software systems require different configurations for every single agent. These two aspects give rise to a mounting complexity of experiment implementation when the number of objects amounts. In this section an algorithm to track all the uavs is developed, basing on an uniform marker configuration used for every uav.

2.1 Method Overview

Based on identical marker configuration whose geometry characteristic is initially known, we present in this section an algorithm to track vehicles. The algorithm fuses the velocity informations to give predictions, in addition to the geometry characteristic and the relative positions of the markers. After each estimation, the algorithm uses the absolute value of the error of the previous estimation to correct the current prediction. The algorithm is offboard and is driven at 50 Hz on a PC by the ROS node who receives vicon marker ROS message.

2.2 Steps in the Method

The algorithm can be divided into several phases:

1. Prediction:
   In the prediction phase it is assumed that the velocity stays constant, and the prediction is modified in accordance the error of the prediction on last time step. The prediction phase is presented in the Eq. (1):

$$\widetilde{x_k} = x_{k-1} + v_{k-1} \times \Delta t + \rho \times \varepsilon_{k-1} \tag{1}$$

2. Registration of markers to corresponding vehicles[1]:
   In this step, it is assumed that changes in attitude between two time steps are negligible and there is no command on yaw angle ψ and its rate ω. Experiments show good performance of this assumption at moderate aggressive flight. When no more than 2 points around the predicted center position of an uav are detected, a most similar frame comparison (MSFC) algorithm is implemented. In order to find out in what direction the detected point is,

Algorithm 1. Marker registration algorithm

Input: Point cloud: vicon markers
Output: Vehicle positions x_k

algorithm to track object from a given point cloud
if the first time detect marker **then**
    Initialization: assign the positions x_0 and yaw angle ψ for every vehicle, save the relative vectors r
else: at step k
    for vehicle i **do**
        find the set of points close to $\widetilde{x_{k,i}}$: $Pts = \{pt_0, pt_1, ..., pt_n\}$
        if $|Pts| > 2$ **then**
            check the geometry relation of the vectors formed by points in Pts
            if right geometry relation **then**
                calculate $x_{k,i}$ using Pts
                find vehicle i, go to the $i+1$ vehicle
            else
                Cannot find vehicle i, go to $i+1$ vehicle
        if $|Pts| <= 2$ **then**
            for every pt_j *in* Pts **do**
                $r_{i,j} = pt_j - \widetilde{x_{k,i}}$
                find the most similar r (note as $r^{\star}_{i,j}$) to $r_{i,j}$
            set the x_k as $\overline{pt_j - r^{\star}_{i,j}}$
        else
            Cannot find vehicle, go to the $i+1$ vehicle
return positions x_k

referring to the predicted center, the MSFC algorithm calculates the vector from the predicted center to the detected point, compares it with the four original vectors registered at initialization step and selects the most similar one as the direction the current detected point is in, relative to the predicted center.

3. Renew the error of the prediction in the step k: in this phase of the algorithm, the error of the predicted position is recalculated and is used to renew the weight it takes in the next estimation step.

$$\varepsilon_k = x_k - \widetilde{x_k} \tag{2}$$

$$\delta = \|\varepsilon\| \tag{3}$$

$$\rho = \frac{\delta}{\delta + \lambda} \tag{4}$$

where λ is a hyper-parameter which depends on the average absolute of the error. We design the expression of the weight ρ as in (4) because it is aimed to adapt the weight of the prediction in accordance to the norm of its error.

2.3 UAV Marker Configuration and Its Initialization

In [12] a method of initialization is mentioned with a guess of the yaw angle for each vehicle. This method requires iterations for the guess of the initial yaw angles begin from which they converge to their estimated values. This method provides possibilities of recognition failure and high computational cost. Within the scope of this paper a square configuration is used as the initialization configuration for uav markers, and the center of the squares represents the position of uavs. For considerations on precision and computational effectiveness, the initial yaw angles are manually assigned for every vehicle through an opencv interface each time the experiment is launched. This assignment successfully initializes the yaw angle of every vehicle in correspond with their true directions related to the coordinate system in the vicon software. With all these informations, the initial position of every vehicle and four vectors pointing from the center to the four markers of the vehicle is obtained. In later discussions, for the purpose of simplification, it is assumed that there is no command on yaw angle and its rate; further, it is also assumed that during the flight the changes of attitudes are negligible.

2.4 Velocity Filtering

As shown in Sect. 4, experiments show that when the rate of marker dropout is high, the estimation of velocities is accompanied with sharply oscillated noises. A filtering solution is proposed that, before it is used for prediction in the next estimation step, a first-order low pass filter with self-adapted parameters is designed and implemented to extract the low frequency component of the raw velocity $\hat{v_k} = x_k - x_{k-1}$ from the noise. The principle of first order low pass filter is presented in (5):

$$v_k = \alpha \times \hat{v_k} + (1 - \alpha) \times v_{k-1} \tag{5}$$

Self-adaptation design of the filter parameter α is based on the principles: (1) The filter is designed to track the input data when it varies rapidly (if the variation of the input data is larger than a threshold ϵ). The agility of the filter augments with the varying rate of the input data. (2) The filter is designed to minimize the impact of sharp changes (if the direction of two consecutive variation changes, the counter and α are reset to zero). (3) The filter is designed to follow the consecutive augmentation or decrease of the input data (if two consecutive changes is towards the same direction, we tend to augment α). Here the design of the self-adapted α is presented in Fig. 1.

3 Control

The controller used in this paper is based on [13], to which augmented a cascade design *(P-PID)* for position control with the inner loop is a proportion controller for position-velocity and the outer loop is a PID controller for velocity-acceleration. Both the inner and outer loop are added with feed forward terms

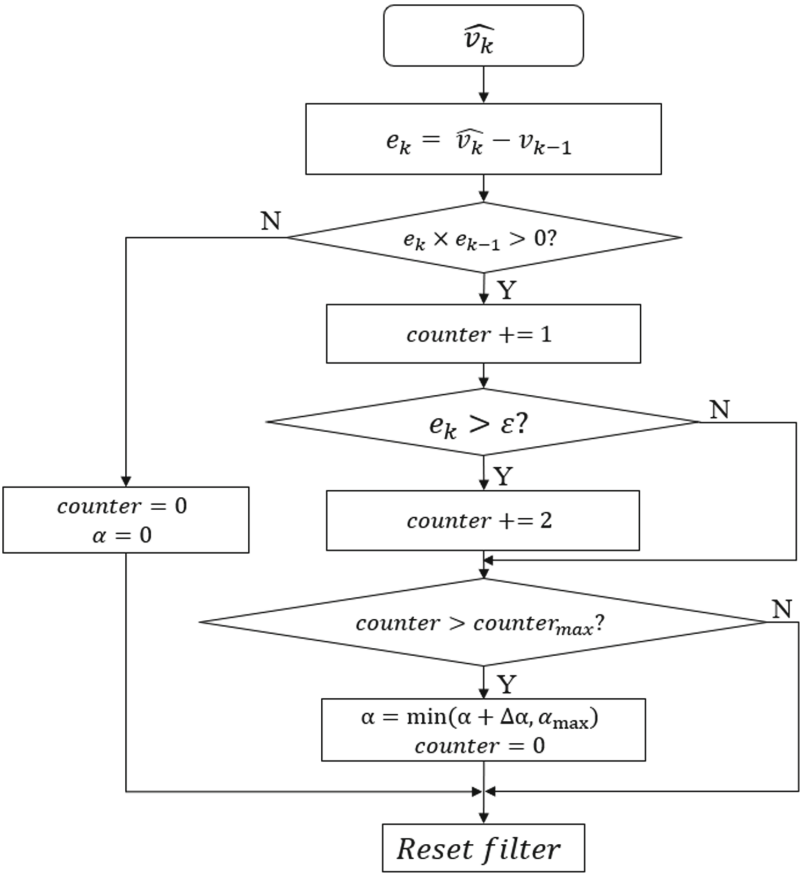

Fig. 1. Design of self-adapted low-pass filter

from the trajectory plan. The on-board control input is the attitude setpoints $(\phi, \theta, \dot{\psi})$, and the off-board commander setpoints are position p_{des}, velocity v_{des}, and acceleration a_{des}. Position error is defined as

$$e_p = p_{des} - p \tag{6}$$

In the inner loop the velocity is renewed with the position error, and is fused with the feed forward velocity from the planned trajectory:

$$\tilde{v} = K_{pp} \times e_p \tag{7}$$

$$v = \tilde{v} \times (1 - \rho) + v_{des} \times \rho \tag{8}$$

In the outer loop we define velocity error as

$$e_v = v - v_{des} \tag{9}$$

and the velocity error is utilized to renew the acceleration with a PID controller, fused with the feed forward acceleration from the trajectory plan:

$$\tilde{a} = K_p \times e_v + K_i \times \int e_v + K_d \times \Delta e_v \tag{10}$$

$$a = \tilde{a} \times (1 - \eta) + a_{des} \times \eta \tag{11}$$

where K_{pp}, K_p, K_i, K_d are positive diagonal matrices.

4 Experiment

All experiments in this paper are conducted with the crazyflie micro uav [14]. A Vicon motion capture system is used for testing the state-estimation method and for estimating vehicles' position and velocity. Onboard gyros and IMU sensors are used for estimating the vehicles' attitude. Our software is written in C++ in the ROS kinetic environment. It is running on a PC with Ubuntu 16.04, TM i7-6700HQ, 2.60 Hz, and 16 GB RAM. We use ROS messages to transmit the point cloud information between motion-capture system and PC, and also the state estimation between PC and UAVs.

An experiment is conducted with the command of two vehicles to execute taking off (two phases: 0.2 m/s and 0.3 m/s) - hovering - circling (two uavs getting to the same radius of 0.8 m and beginning to circling at angular rate of 0.785 rad/s (equivalent to 8 s/r)). The experiment is conducted with the same command under different conditions: one with low marker dropout rate and the other with high marker dropout rate.

A. Performance without marker dropout:

In optimal conditions where marker dropout rate stays nearly to zero, the algorithm presented performs well in the test of circling as shown in Fig. 2. The results show that the position of the two vehicles follows well the curves of sinusoid and the velocities without much oscillations.

B. Velocity filtering in presence of data dropout and the position estimation result.

It is shown in Fig. 2 that the estimation of velocity is obtained with oscillations and noise. Experiments were conducted to try to find out the relation between noise presented in velocity and data dropout rate. In some extremely noisy conditions where the estimated velocity of vehicles oscillates violently added with marker dropout rate staying at a high level, vehicles are easily out of control even when they are commanded to track simple trajectories. Figures 3 and 4 show the impact of marker dropout rate on the oscillation of estimated velocity. When the two vehicles were commanded to follow the same trajectory with the same angular speed, vehicle0 who flies under severe marker dropout condition has considerable oscillations in the velocity curve while the other vehicle gives back tolerable estimation of velocity. In order to augment the precision of the position prediction, we implement the Low-Pass filter to the velocity calculated by difference of two consecutive positions, as mentioned in Sect. 2.4.

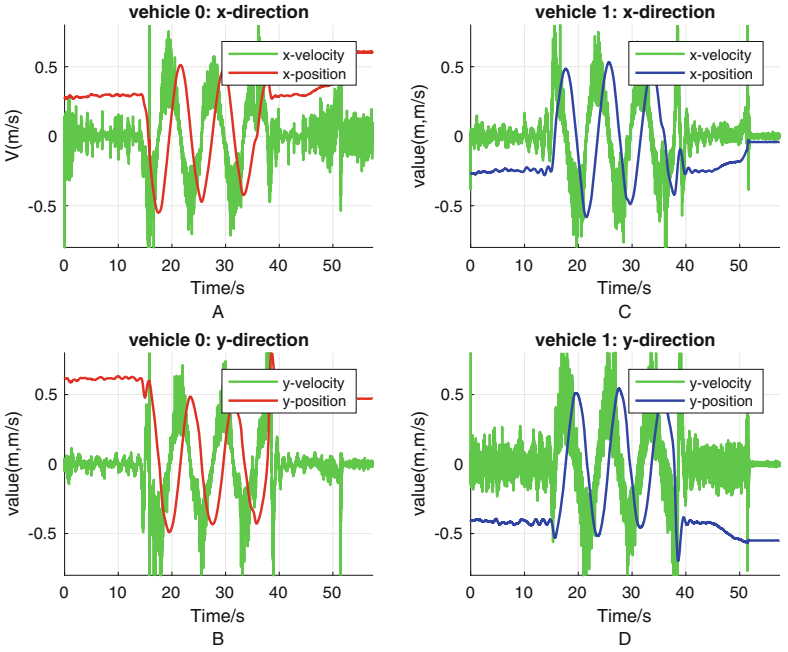

Fig. 2. Position of two vehicles commanded circling

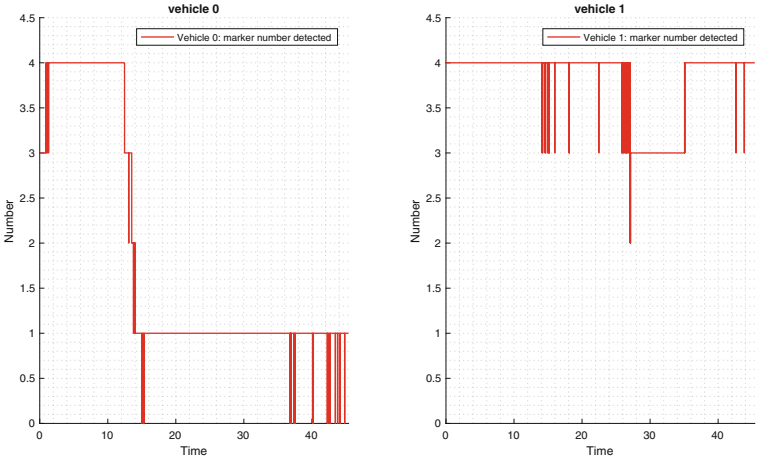

Fig. 3. Number of belonging markers detected for each of the two vehicles

From Figs. 3 and 4 we conclude that while different rates of marker recognition affects the calculated value of velocity, we can reduce the violent and sharp oscillation of velocities by filtering it with the filter proposed. State estimation with velocity filter added in the prediction step. Figure 5 shows the result of position estimation after filtering the velocity.

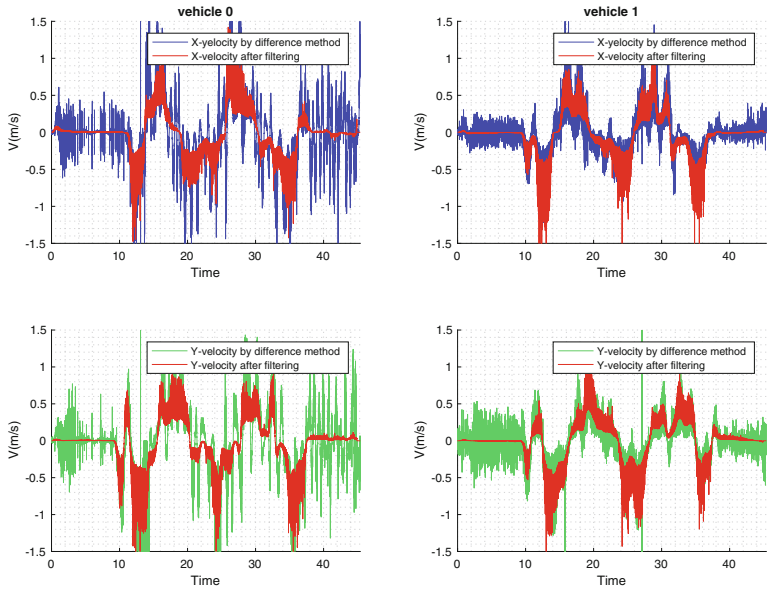

Fig. 4. Velocity oscillations under different data-dropout conditions

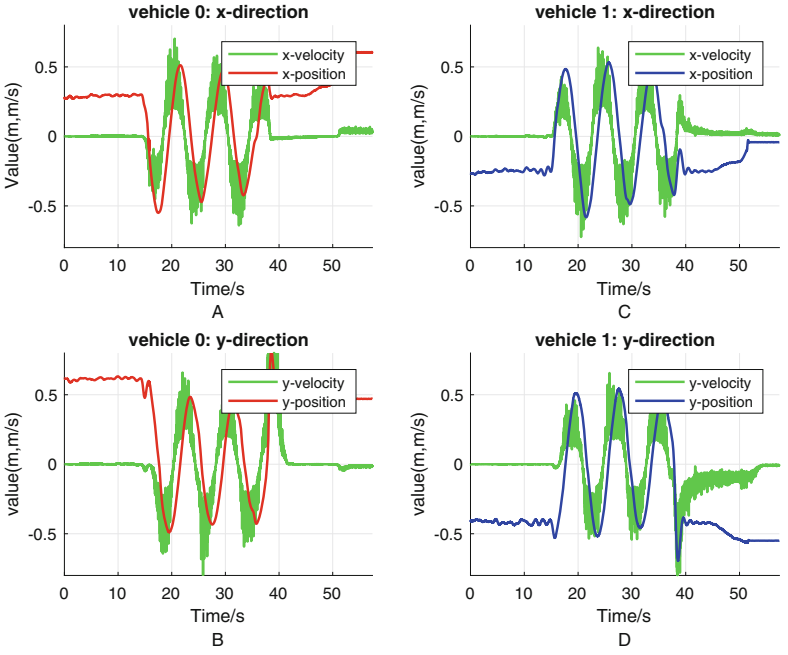

Fig. 5. Position and velocity of two vehicles after passing the low pass filter

5 Conclusion

State estimation is the base and one of the key components in robot control systems. We have described a method for indoor state estimation of swarm uavs under influences of random noise and data dropout. In this paper, the state estimation issue is simplified by utilizing the high precision of measurements provided by the motion capture systems. Although indoor motion capture systems provides high resolution in single-point position measurements, our algorithm takes into consideration complex situations met by swarm robot control in terms of state estimation and object tracking. To achieve robust performances, we fully utilize the geometry characteristic of the markers on one vehicle, together with precise prediction of positions based on smooth velocities filtered by a self-adaptive low pass filter. Experiments show good tracking and filtering results. The method mentioned in this paper is hoped to serve as the testbed for more complicated swarm robotic control systems in the future. In future work, it is planned to include the control on yaw angle and its rate into the prediction of the position, together with other attitude changes, in order to augment the capability and robustness of the algorithm in more aggressive flights.

Acknowledgement. This work was partially supported by the National Natural Science Foundation of China (Grant No. 51605282) and National Science and Technology Major Project.

References

1. Giles, K., Giammarco, K.: Mission-based Architecture for Swarm Composability (MASC). Procedia Comput. Sci. **114**, 57–64 (2017)
2. Yatskin, D., Kalinov, I.: Principles of solving the space monitoring problem by multirotors swarm. In: International Conference on Engineering and Telecommunication (EnT), pp. 47–50 (2017)
3. Schneider, D.: Open season on drones? IEEE Spectr. **51**(1), 32–33 (2014)
4. Tanzi, T., Apvrille, L., Dugelay, J.L., et al.: UAVs for humanitarian missions: autonomy and reliability. In: Global Humanitarian Technology Conference, pp. 271–278. IEEE (2014)
5. Loayza, K., Lucas, P., Pelaez, E.: A centralized control of movements using a collision avoidance algorithm for a swarm of autonomous agents. In: IEEE Second Ecuador Technical Chapters Meeting, pp. 1–6. IEEE (2017)
6. Albani, D., Nardi, D., Trianni, V.: Field coverage and weed mapping by UAV swarms. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4319–4325. IEEE (2017)
7. Arpino, G., Morris, K., Nagavalli, S., Sycara, K.: Using information invariants to compare swarm algorithms and general multi-robot algorithms: a technical report. In: IEEE International Conference on Robotics and Automation (2018)
8. Wan, S., Lu, J., Fan, P.: Semi-centralized control for multi robot formation. In: International Conference on Robotics and Automation Engineering (ICRAE), pp. 31–36 (2017)

9. Xuan, P., Lesser, V.: Multi-agent policies: from centralized ones to decentralized ones. In: International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1098–1105. ACM (2002)
10. Lupashin, S., Hehn, M., Mueller, M.W., Schoellig, A.P., Sherback, M., D'Andrea, R.: A platform for aerial robotics research and demonstration: the flying machine arena. Mechatronics **24**(1), 41–54 (2014)
11. Ledergerber, A., Hamer, M., D'Andrea, R.: A robot self-localization system using one-way ultra-wideband communication, Mechatronics, In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3131–3137 (2015)
12. Preiss, J.A., Honig, W., Sukhatme, G.S., Ayanian, N.: Crazyswarm: a large nano-quadcopter swarm. In: IEEE International Conference on Robotics and Automation, pp. 3299–3304 (2017)
13. Mellinger, D., Kumar, V.: Minimum snap trajectory generation and control for quadrotors. In: IEEE International Conference on Robotics and Automation, pp. 2520–2525 (2011). https://doi.org/10.1109/ICRA.2011.5980409
14. Bitcraze. https://www.bitcraze.io/. Accessed 4 Oct 2017