STRATEGIES FOR TEAMING OF UNMANNED AERIAL VEHICLES

J. Seibold, O. Meister, N. Frietsch, and Prof. Dr. G. F. Trommer
Institute of Systems Optimization
University of Karlsruhe (TH)
Kaiserstr. 12, 76128 Karlsruhe

ABSTRACT

This paper focuses on strategies for teaming of vertical take-off and landing micro aerial vehicles (VTOL-MAV) based on four-rotor helicopters. Different team applications are implemented, leading to a cooperation of several autonomous quadrocopters. Two different strategies are presented in this paper. The first possibility is to build formations. Several MAVs build a leader-following formation in any shapes. The second strategy for cooperations of the team is the synchronization during waypoint flights. To compensate for time gaps occuring due to disturbances like wind, it is useful to install waypoints that provide synchronization.
To implement all such features a digital radio link is needed. The links are restricted in terms of weight, size and power consumption because of the installation on the MAV. The realization of the radio link and the used protocol are described.
Finally, the results of the implementation are presented. In a first step the algorithms are tested in a simulation using software-in-the-loop. In order to validate these results, field tests have been realized with two drones.

1. INTRODUCTION

In the last few years there have been more and more research activities in the field of unmanned aerial vehicles (UAV). Large UAVs can carry high payload and have large operating distances. By contrast there are small UAVs with smaller payload and range, but lower costs and easier deployment. Such UAVs are used in several applications like surveillance and reconnaissance missions, which require the ability to hover. The UAV used for this application is a four-rotor, electrically powered helicopter (shown in Figure 1), with hovering abilities. This vertical take-off and landing micro aerial vehicle (VTOL-MAV) has features to perform autonomous waypoint navigation [1].
With the help of geographical information systems (GIS), like Google Maps [3], waypoint-routes are planned on the PC. The command-based architecture allows the user to build waypoint-routes with different operations like positions changes and payload action. The synchronization of several drones during their waypoint flights is an approach to expand the functionalities of the waypoint navigation. Due to disturbances, like wind, it is not possible to plan parallel routes for many drones. A decoupling in time would occur.

Figure 1: microdrones md4-200 [2]

The field of formation flying has seen many researches in the past: [4–7]. Many papers deal with leader-following formations. In this case a leader is assigned to every agent in the formation. To obtain the formation, the control of the position of the agents

is realized in two different ways. If only one leader is defined, the agent controls the heading ψ and the distance l to its leader ($l - \psi$ control). In the case of two defined leaders, the two distances to the leaders are the controlled variables ($l - l$ control). Algorithms to achieve these control goals are discussed in [4–6]. [7] describes different frames of reference. A first approach is to assign the formation-leader to every agent in the team as its own leader. All agents control their position relative to this reference.
Another approach assigns neighbors as the leader. In this case, the position of the leader can be obtained easier, because in most cases the neighbor is in sight and can be detected with a camera. Such an approach implies very low communication effort, as no positions have to be broadcasted.
The third approach is to define a unit-center as the average of the positions of all roboters in the formation. Every agent gets a position assigned relative to this center.
In all these cases, the communication between the agents of the team is very low. Only during the initialization the positions of every roboter has to be defined. During the formation flight, the agents obtain their own position from the position of their leader and the lever arm. Local sensors can easily detect the position of the leading roboter.
In this work, a leader-following formation is realized. All following agents are assigned to the formation-leader. To calculate their own positions, the agents need to know their lever arm l_i and the position of the master $\vec{x}_{master}$. The lever arm can be defined in the body- or the navigation-frame. This allows very varied controls of a formation.

This paper is organized as follows. Section 2 describes the chip-select and the integration of the radio link. This section also introduces the implemented protocol. In section 3 the implementation of the formation flying is discussed, followed by the implementation of the waypoint-synchronization in section 4. To validate and test the algorithms, a software-in-the-loop environment has been created, and is discussed in section 5. The performance of the integrated algorithms is shown in section 6 with the help of both simulation and field test results, and finally conclusions are drawn.

2. IMPLEMENTATION OF THE RADIO LINK

In this section the implementation of the radio link is described. In a first step, a radio module has to be chosen. Based on this communication link a protocol is designed to provide the performance of the team algorithms.

2.1. Radio Module

To realize the team strategies, a digital communication link between the drones and an operator is needed. As the radio modules have to be installed on the UAV there are physical restrictions in terms of weight, size and power consumption. As the md4-200 has a diameter less than $1\ \mathrm{m}$, the size of the radio link is restricted. The weight of $800\ \mathrm{g}$ and the maximal take-off weight of $1\ \mathrm{kg}$ are leading to the restriction in terms of weight. This $200\ \mathrm{g}$ payload is needed for sensors of any kind, such as optical sensors like cameras. The aspect of the low power consumption is in conflict with the demand of high range. Due to the limited battery capacity all power consumers on the quadrocopter decrease the duration of flight.
To ensure the performance of the radio link, the range, transfer rate and the robustness are important demands for the radio modules. High range and high transfer rate are in conflict with the power consumption, so a compromise had to be found. The robustness of a radio link describes the possibility to reconnect easily after the connection failed.
Several wireless technologies have been surveyed. *Digital Enhanced Cordless Telecommunication* (DECT) is used in cordless phones. For this reason there were no modules available, that achieve all given constraints.
The *bluetooth* technology is intended for high-speed data links. The purpose of bluetooth connections is to ensure high data rates over distances about $30\ \mathrm{m}$ Some bluetooth modules reach up to $1000\ \mathrm{m}$ line-of-sight, which is still too little for this application.
The final choice is the radio module *TinyOne Pro* [8], which uses the $868\ \mathrm{MHz}$-ISM band[1]. The range of $4\ \mathrm{km}$ and the transfer rate of $38.4\ \mathrm{kbps}$ form an acceptable compromise. As the connection between the modules does not have to be established, the communication link is very robust. The module is connected to the flight controller micro controller over the UART interface[2], which makes the handle of the communication easy.

2.2. Protocol

A protocol is designed to ensure the correct communication between all team members. The layout of the protocol is as following:

1. message header
2. serial number
3. data
4. checksum

[1] **I**ndustrial, **S**cientific and **M**edical band; ISM bands are license-free radio bands, that can be used for industrial, scientific and medical purposes
[2] **u**niversal **a**synchronous **r**eceiver **t**ransmitter

The $2\ \mathrm{byte}$ message header is subdivided in message class and message ID. The message class divides the messages into three parts:

- inflight
- set-up
- teaming

Messages of the classes *inflight* and *teaming* are only parsed during flight, *set-up*-messages only during the pre-flight phase.
The message ID specifies the goal, so that the *data* is parsed correctly. The transferred serial number defines the receiver of the message. In some cases broadcast messages are useful, for example to send DGPS corrections[3]. These broadcast messages can be handled with a zero as the serial number of the receiver. The checksum at the end of every message ensures a correct transfer.

3. FORMATION FLIGHT

This section contains all functions implemented to perform team activities. Starting with the team acquisition, the leader (master) has to acquire the required slave drones. During this acquisition several conditions are reviewed by the slave drones. To perform the formation flight, the lever arms have to be assigned to all slaves. They can be defined relative to the body- or to the navigation-frame. The release of the slaves is executed at the completion of the mission.

3.1. Acquisition

As a first approach, the formation flight is implemented as a centralized, distributed system. This means that the intelligence of the system is focused in one master drone. This master initializes the acquisition of the required slaves. A request is performed by sending a broadcast message with the serial numbers of the desired slaves. After parsing this message, all drones decide if they join the team or not. It is checked if the drone is available to join a team, or if it is already in another team. If it is available, an acknowledgement message is sent to the requiring master drone. Otherwise the request is rejected as not acknowledged. Due to this handshake procedure the master drone always knows its number of slaves and their identities.

3.2. Lever arm

The formations are defined on the basis of the lever arms l_i. The lever arms can be defined in two different ways to offer a variety of options. If the lever arm is defined in relation to the navigation frame, the position of the slave drone is defined as *north*, *east* and *down*. This is an easy way to implement simple formations. The second possibility is to define the lever arm in relation to the body-frame of the master drone. In this scenario the slave drone's position can be $10\ \mathrm{m}$ behind, $5\ \mathrm{m}$ to the left and $2\ \mathrm{m}$ higher than the master drone. In this case, the guidance of the slave drone is more complicated, as yaw movements of the master drone result in circular movement of the slave drone. The options to define the lever arms

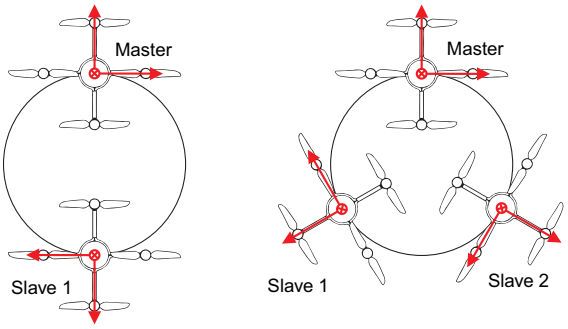


Figure 2: possible formations for surveillance tasks

gives a very broad range of building formations. Figure 2 shows formations useful for surveillance tasks, because the cameras cover most terrain.
A completely different model for a formation flight is to build up a relais station. A slave drone can be positioned right between the user and the operating drone to double up the operation range. In this case, the lever arm has to be defined in the navigation-frame. The lever arm has to be updated once in a while to ensure the right position of the slave, if the operating drone is moving.

3.3. Guidance

The guidance of the master drone can be controlled in two ways. In a first way the drone can be piloted with the radio control. The operator pilots the master drone in the regular manner. The other possibility is to guide the master drone with the waypoint navigator. A predefined waypoint route will be established by the master drone. The advantage of the waypoint-based guidance is that the flights produce repeatable results. The guidance of the slaves drones is performed by the teaming module.
The slave drones calculate their set positions on the basis of the lever arm l_i and the position of the master drone $\vec{x}^e_{master}$.

$$\vec{x}^n_{master} = \mathbf{C}^n_e \cdot (\vec{x}^e_{master} - \vec{x}^e_{POFF}) \tag{1}$$

$$\vec{x}^n_{set} = \vec{x}^n_{master} + \vec{l}_i \tag{2}$$

$$\vec{x}^n_{set} = \vec{x}^n_{master} + \mathbf{C}_\psi \cdot \vec{l}_i \tag{3}$$

[3] **D**ifferential GPS, DPGS corrections can eliminate common errors

$\vec{x}^n_{master}$ represents the position of the master drone in the slave navigation-frame. The local navigation-frame is defined with the help of the first GPS position $\vec{x}^e_{POFF}$. (2) refers to a navigation-frame based lever arm, (3) to body based lever arm. $\mathbf{C}_\psi$ handles the rotation according to the yaw angle of the master drone. This simple approach of the slave guidance leads to "digital" behavior, as the update rate of the set position is only a few hertz. When a new set position is calculated, the drone tries to reach this point at full throttle. Once the point is reached, the drone stops and waits for a new set position. This approach leads to an oscillation which is not acceptable, because it makes surveillance tasks more difficult.
The solution to this problem is to also transmit the velocity of the master drone. Now the set position of the slave drone can be smoothed using the velocity and direction:

$$\vec{a} = \vec{x}^n_{set} - \vec{x}^n \tag{4}$$

$$\tilde{\vec{x}}^n_{set} = \vec{x}^n + v_{abs} \cdot \Delta t \cdot \frac{\vec{a}}{|\vec{a}|} \tag{5}$$

The calculation (5) is executed in every main-loop of the slave drone, so that there are only smooth transitions in the set position $\tilde{\vec{x}}^n_{set}$.

There are more calculations to be done in case of a body-frame based lever arm. As described in section 3.2, the yaw movement of the master results in circular movements of the slaves. To solve this problem, the tangential track speed is calculated for the slave drone. Otherwise, the slave drone would try to reach the set positions at a very low speed, as the master drone is hovering. The use of the calculated tangential track speed prevents from the decoupling between real and set positions.

$$|\vec{v}^n_{track}| = \dot{\Psi} \cdot l_i \tag{6}$$

$$\tilde{\vec{x}}^n_{set} = \vec{x}^n + |\vec{v}^n_{track}| \cdot \Delta t \cdot \frac{\vec{a}}{|\vec{a}|} \tag{7}$$

In (6) $\dot{\Psi}$ represents the angular speed of the yaw movement of the master drone.

4. WAYPOINT SYNCHRONIZATION

As described above, it is a challenge to schedule parallel waypoint routes for multiple UAVs. As there are disturbances like wind, the actual speed of the drone is not always the defined speed. This results in a decoupling of the waypoint routes.
This problem can be solved out trough digital communication. At every point of the waypoint list a synchronization message can be placed. Such a message makes the drones wait. In a first approach a drone can be synchronized with up to three other drones.

4.1. Mission planning

Some aspects have to be considered in the pre-flight phase to have the synchronization working well during the flight.

1. There is no concrete acquisition needed for the synchronization.
2. Only the serial numbers of the involved UAVs have to be known.
3. To rule out the possibility of hanging in the synchronization due to no respond of the partners, a time-out value has to be defined.

4.2. Synchronization Process

The following aspects have to be considered to ensure a fail-safe synchronization process:

- wait for all listed drones
- bridge radio link blackouts
- install a fail-safe routine.

The process of synchronization is realized trough a handshake procedure. Every drone that receives a synchronization order sends a broadcast message including the serial numbers of all drones it is waiting for. After sending this message, the drone enters and stays in a wait-for-synchronization mode. In this mode, the drone checks all synchronization messages to see if it was sent by a drone it is waiting for. In this case, it deletes the corresponding drone from its waiting list and sends the updated synchronization message. This procedure guarantees that all involved drones are considered in the synchronization.
As a radio link blackout can damage synchronization messages, it is not guaranteed that all messages are transferred. To eliminate these failures as a security feature, the actual waiting lists are sent periodically. This ensures that dropped messages will be repeated, and the handshake will succeed.
Another possible problem is, that for some reasons a drone involved in the synchronization is not available. In this case, the partner drones would infinitely remain in the wait-for-synchronization mode. The programmed waypoint route could not be finished as purposed. This issue can be fixed by installing a time-out value. When entering the wait-for-synchronization mode, a timer is started and will terminate the synchronization after the time-out is reached.

A synchronization with complete teams is possible as the formations are centralized & distributed systems. It is enough to include the master of a formation in a waiting list. All the slave drones will be synchronized as well, as they receive their set positions from the master drone.

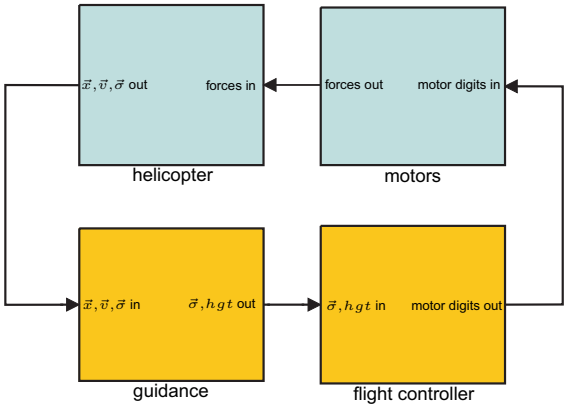


Figure 3: Simulation: block diagram

5. SIMULATION

Testing new algorithms direct on the target hardware is costly in terms of time and can cause crashes of the helicopter. Therefore all implementations have been tested in a software-in-the-loop environment. The environment simulates only the hardware parts for the helicopter. This includes the motors, rotors and aerodynamics. A block diagram of the simulation architecture is shown in figure 3. The software parts are directly taken from the helicopter software. The *guidance* and the *flight controller* blocks contain the original program code, as used on the destination hardware. This implies, that new algorithms can be tested very fast and without any risk of crashes.
To provide the option of testing team functionalities, two drones are implemented in the simulation including a simulation of the communication link. The radio link is assumed to be ideal, so that there are no transfer errors and timing issues.

5.1. Simulation: Guidance

To ensure that the simulation can be used as a software-in-the-loop environment, there is original program code integrated. The block *guidance* provides the same calculations for the desired angles and height as the microcontroller performs on the helicopter. Therefore the actual position and the desired positions have to be accessible. The actual position is provided by the block *helicopter*, as shown in section 5.4.

5.2. Simulation: Flight controller

The block *flight controller* is built up similar to the block *guidance*. Original program code is also used in this instance to determine the set value of the motors. In this block the hover point is computed too. This is the set value of the motors to hold the helicopter in the same height.

5.3. Simulation: Motors

In the *motors* block, the simulation of the real motors including the rotors is implemented. According to a non-linear characteristic curve, the motor set values are computed in separate lift forces.

5.4. Simulation: Helicopter

The computed lift forces are used in the block *helicopter* to propagate the position and attitude of the helicopter. The rotations are caused from force differences; differences between the front and rear rotor cause pitch rotations, between left and right roll rotations. Yaw rotations arise from differences between clockwise and counter-clockwise rotating rotors.
The calculated state of the helicopter is passed to the *guidance* to close the loop.

6. RESULTS

Several tests have been passed to validate all implemented algorithms. First, as described above, the simulation has been used. To ensure that the simulation results are suitable there has been a comparison of the simulation and real flights. Second, the team functionalities have also been tested on the devices, using two drones.

6.1. Comparison: Simulation vs. real flight

The behavior of the simulation can be tested with a waypoint flight which a real drone had been flying before. The logged data of the real flight and the simulated data can be used to validate the software-in-the-loop environment.
The performed waypoint route is a path planned at the University of Karlsruhe with a duration of about ten minutes. In the next plots, the important parameters of the flights are compared.

6.2. Teaming Results

The team performances have been tested on two drones, one master and one slave drone. The result plots visualize the positions, velocities and attitudes of both drones.
As shown in figure 5d there is a time delay in the velocities. As the master drone sends its position and velocity with a fixed frequency, the slave drone increases its velocity later as the master drone. Therefore, the slaves flies a little longer than the master, to reach the right final position

7. CONCLUSION

This paper describes the development and implementation of team algorithms for a small unmanned four-rotor helicopter. The implementation of a radio link and a communication protocol was realized to provide the basis for the team applications. Different team strategies have been presented and validated.
The formation flight was implemented with very flexible formation-structures. Due to the possibility of defining the lever arms in different frames, the versatility is very high.
The synchronization mode during waypoint flights allows to plan parallel routes on different UAVs. Disturbances, like wind, are compensated at the synchronization points.
A software-in-the-loop environment has been developed to simplify the cycle of developing and testing. Based on this simulation tool the implemented functions could be easily verified without the risks associated to a real hardware implementation.
Both the simulation and the field test results showed successful team strategies for VTOL-MAVs.
Further work will include other types of autonomous vehicles, like unmanned ground vehicles as well as the change to a decentralized, more flexible architecture.

ACKNOWLEDGEMENTS

The authors wish to thank the microdrones GmbH.

References

[1] O. Meister, C. Schlaile, R. Monikes, J. Wendel, and G. Trommer, “Development and validation of a vtol mav waypoint navigator,” *SYMPOSIUM GYRO TECHNOLOGY*, p. 12, 2006.

[2] microdrones GmbH, 2008. [Online]. Available: http://www.microdrones.com/

[3] GoogleMaps, 2008. [Online]. Available: http://www.maps.google.de/

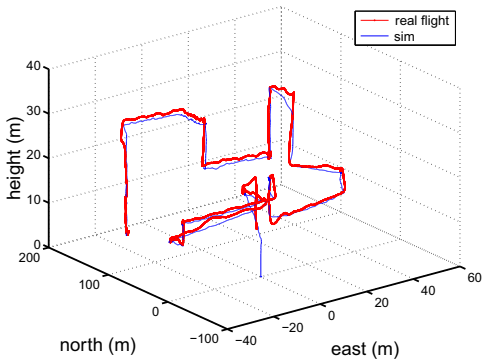
[4] J. Shao, G. Xie, and L. Wang, “Leader-following formation control of multiple mobile vehicles,” *Control Theory & Applications, IET*, vol. 1, no. 2, pp. 545–552, 2007.

[5] J. Fredslund and M. Mataric, “A general algorithm for robot formations using local sensing and minimal communication,” *Robotics and Automation, IEEE Transactions on*, vol. 18, no. 5, pp. 837–846, 2002.

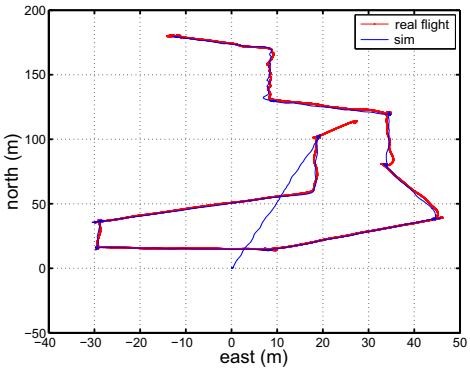
[6] P. Kostelnik, M. Samulka, and M. Janosik, “Scalable multi-robot formations using local sensing and communication,” *Robot Motion and Control, 2002. RoMoCo’02. Proceedings of the Third International Workshop on*, pp. 319–324, 2002.

[7] T. Balch and R. Arkin, “Behavior-based Formation Control for Multi-robot Teams,” *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 926–939, 1998.

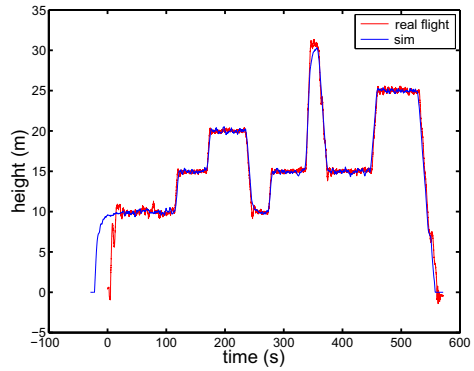
[8] *Data sheet: TinyOne Pro 868 MHz OEM RF module*, One RF Technology, France, 2007.



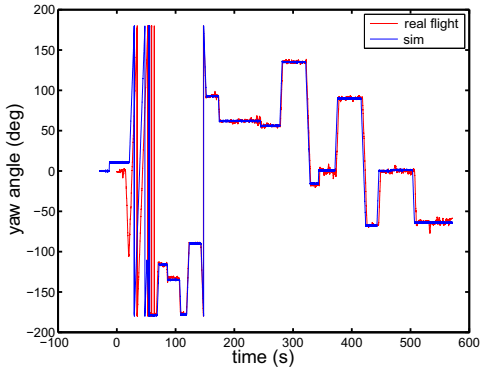
(a) 3D view



(b) top view

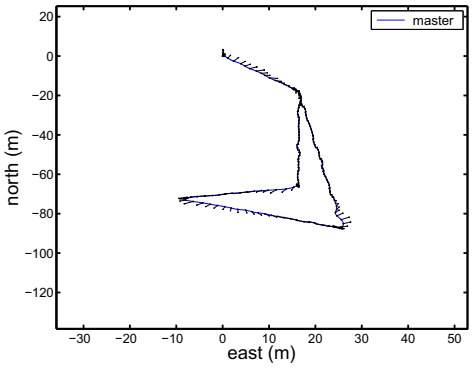


(c) height profile

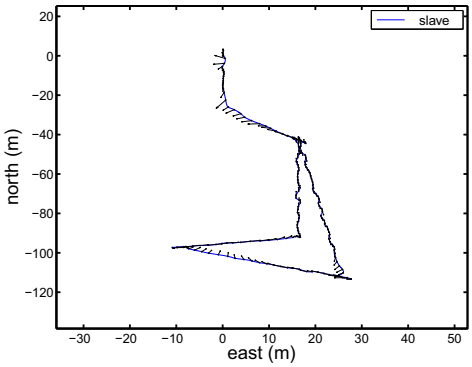


(d) yaw angle

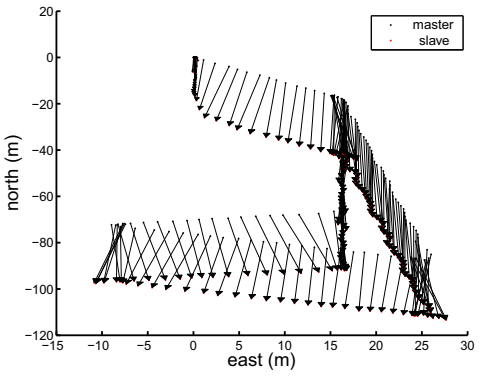
Figure 4: comparison of the simulation with real-flight results



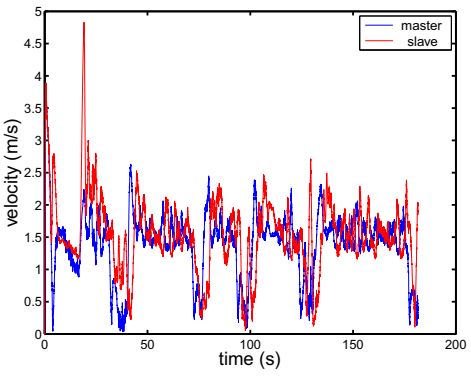
(a) position master



(b) position slave



(c) relation of the positions



(d) velocities of master and slave drone

Figure 5: formation flight with navigation-frame based lever arm $\vec{l}^{n} = (25.0; 0.0; -5.0)^{\mathrm{T}}\ \mathrm{m}$