CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

BACHELOR'S THESIS

Tomáš Velecký

Cooperative Filming of a Moving Ground Object by a Group of Unmanned helicopters

Department of Cybernetics

Thesis supervisor: **Ing. Viktor Walter**

May 2018

Author statement for undergraduate thesis

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Kongens Lyngby, 11th May 2018 .....

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Velecký Tomáš** Personal ID number: **457241**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

Branch of study: **Robotics**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Cooperative Filming of a Moving Ground Object by a Group of Unmanned Helicopters

Bachelor's thesis title in Czech:

Kooperativní filmování pozemního pohybujícího se objektu skupinou bezpilotních helikoptér

Guidelines:

The goal of the thesis is to design, implement in ROS (Robot Operating System), and experimentally verify in Gazebo simulator an algorithm for filming a visually-localized object by a formation of mutually stabilized unmanned aerial vehicles (UAV). The following tasks will be solved:
1. To implement a method [1], which uses distributed vision-based flying cameras to film a moving target, and integrate it into the UAV control system of the MRS group at CTU.
2. To use a method of visual detection of objects denoted by an artificial pattern [2,4] for detection of the moving target and for mutual localization of neighboring vehicles.
3. To extend the method in [1] by considering the knowledge of the ground profile measured by distance sensors onboard of UAVs and by considering the complete state of the UAVs into the control rules.
4. To verify the method and compare it with original approach in a realistic Gazebo simulator.
5. To prepare the system for experimental verification with the multi-UAV platform of the MRS group [4] (the real experiment will be realized in case of available HW based on decision of the thesis advisor).

Bibliography / sources:

[1] F. Poiesi and A. Cavallaro. Distributed vision-based flying cameras to film a moving target. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015.
[2] J. Faigl, T. Krajník, J. Chudoba, L. Preucil, and M. Saska, "Low-cost embedded system for relative localization in robotic swarms," in Proc. of IEEE International Conference on Robotics and Automation, 2013.
[3] T. Krajnik, M. Nitsche, J. Faigl, P. Vanek, M. Saska, L. Preucil, T. Duckett and M. Mejail. A Practical Multirobot Localization System. Journal of Intelligent & Robotic Systems 76(3-4):539-562, 2014.
[4] T. Baca, P. Stepan and M. Saska. Autonomous Landing On A Moving Car With Unmanned Aerial Vehicle. In The European Conference on Mobile Robotics (ECMR), 2017.

Name and workplace of bachelor's thesis supervisor:

Ing. Viktor Walter, Multi-robot Systems, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **16.01.2018** Deadline for bachelor thesis submission: **25.05.2018**

Assignment valid until: **30.09.2019**

Ing. Viktor Walter	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Ing. Pavel Ripka, CSc.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. _____
Date of assignment receipt

_____
Student's signature

Acknowledgements

I would like to thank many people in the MRS group for the online technical support, my Czech supervisor, my "Danish supervisor" Nils Axel Andersen, my parents, and the European Union, who all made it possible to do the thesis in the Kingdom of Denmark.

Abstract

This thesis implements an algorithm for vision-based ground target following by a formation of unmanned aerial vehicles described by F. Poiesi and A. Cavallaro [1]. Various modifications were made to account for the specific properties of the platforms used by the Multi-robot Systems (MRS) group [2]. These changes comprised of adjustment of the outputs to the setpoint-based control scheme and high processing speed of the platform. Additionally methods for suppression of formation oscillations and temporary loss of line-of-sight with the target. Another contribution of the thesis is the development of a method for estimation of the slope of the ground on which the target is positioned, that allows for refinement of the target position on a natural terrain. These algorithms were implemented as a node for the Robot Operating System, in order to ensure compatibility with other subsystems used by the MRS. The implemented algorithms were tested in a simulation environment and partially in real-world experiments with physical UAVs.

Keywords: ground object, object following, target, formation, swarm, whycon, ROS, UAV, unmanned helicopters

Abstrakt

Tato bakalářská práce implementuje algoritmus pro sledování pozemního cíle formací bezpilotních helikoptér (UAV) vybavených běžnou kamerou popsaný F. Poiesim a A. Cavallarem [1]. Byly provedeny úpravy umožňující využití helikoptér skupiny Multi-robotických systémů (MRS) [2]. Jedná se zejména o úpravu výstupu algoritmu pro řízení na relativní pozici a pro vysokou výpočetní frekvenci platformy. Dodatečně byly přidány nástroje pro potlačení oscilací a práci s dočasnou ztrátou cíle. Byla vyvinuta metoda pro odhad náklonu povrchu pod formací, umožňující zpřesnění pozice cíle umístěného v přirozeném prostředí. Zmíněné algoritmy byly implementovány jako node pro Robot Operating System (ROS), aby byla zaručena kompatibilita s ostatními subsystémy užívanými skupinou MRS. Algoritmy byly odzkoušeny v simulátoru Gazebo a částečně pomocí experimentů s reálnými helikoptérami.

Klíčová slova: pozemní objekt, sledování objektu, cíl, formace, swarm, whycon, ROS, UAV, bezpilotní helikoptéry

Contents

List of Figures

1 Introduction

This thesis is based on the article *Distributed vision-based flying cameras to film a moving target* by Fabio Poiesi and Andrea Cavallaro from 2015 [1]. The authors propose an algorithm for following a moving target by a group of unmanned helicopters. It is assumed that each helicopter is equipped with a single camera and a detection system, so that the coordinates of the target in the camera image are known. The unmanned aerial vehicles (UAVs) in the formation are also assumed to know the relative positions of their closest neighbours. The main advantage of using more than one UAV is the extention of the area covered by cameras of the formation. If some of the UAVs are unable to detect the target, it is still not lost by the whole formation.

The original approach is based on following these restrictive assumptions: the target is always moving in a plane, at a constant altitude; all the UAVs with the cameras are also at a constant altitude. In the initial state, all the UAVs are able to detect the target; the environment is without obstacles, the ground is flat; and each UAV can communicate only with two of its neighbours. The algorithm does not aim to cope with air disturbations (wind).

The algorithm was already proven in Matlab by the authors with 6, 8, 10, and 12 UAVs in a formation [1]. In this thesis, the algorithm was tested in the Gazebo simulator. This implementation aims to be compatible with the UAVs currently used by the MRS group.

1.1 Implementation Specifics

The algorithm is based on relative localization of the UAVs in a formation. However, the simulation system used by Multi-robot Systems group provides absolute coordinates. The real-world platform used by the MRS group relies on Global navigation satelite system (GNSS) real-time kinematic (RTK) for the control of the UAVs movement. It is therefore possible to test the algorithm with data from an odometry system. Moreover, as the program is only allowed to send absolute or relative *positions* to the model predictive controller (MPC) at this level of control without a model of the UAV, it would not be possible to test the programme completely without RTK.

The program is able to process the data from odometry, completely simulating both the target detection and relative localization of neighbouring UAVs using the Whycon[1] system. It is also able to work with the target coordinates provided by a Whycon node but to use the odometry data for relative localization of other UAVs, at the same time. Both methods of target detection were tested experimentally – using one UAV as a target and another one UAV as the follower, and an unmanned ground vehicle (UGV) with a Whycon pattern fol-

[1]For information about Whycon system, see [3], [4], [5], [6].

lowed by a UAV. A system for relative localization of other UAVs using a Whycon pattern detection was only succesfully tested in the simulator.

The algorithm from [1] expects a control system able to set the relative position of a UAV in a very short time horizon (the authors used 0.04 s in simulations), whereas the UAVs of MRS group can only be controlled by setpoints or sequences of setpoints with 0.2-second steps. This is addressed in section 2.2. Other authors propose algorithms taking data from target detector, directly producing more low-level setpoints (eg. yaw, pitch, roll and altitude in [7]).

1.2 Related Works

Another system for following a moving target was presented in an article from 2012. Only a single UAV was used, with the camera not being fixed and with a known distance to the target. They proposed a two-layer framework to control a pan/tilt servomechanism of the camera to keep the target in the centre of the image and to follow the target by the UAV using a vision feedback [8]. Another article [9] also uses pan/tilt camera mechanism.

Several articles aim to keep the target in the center of the camera image, either by controlling the rotation of the camera ([8], [9], or [10] in the case of a UGV), or directly by controlling movement of the whole UAV ([1] and [7]).

The approach in [7] is not based only on the knowledge of the coordinates of the target in the camera image. A proportional integral controller (PI) was used for movement control, with "detection size, out of plane rotation and estimated object position" as its inputs.

The article [11] concernes with the controller for the UAV position in the x-y plane. Both [1] and [11] were verified in a MATLAB simulation only.

In [12], the authors also use a regular polygon formation shape, while assuming a limited field of view (FOV) of cameras, and their "robots rely only on their onboard visual sensor without external input". In addition to assumptions in [1], they assume *unicycle kinematics* for both the UAVs and the target.

In [13], a gimbal camera and an extended Kalman filtering based target state estimator is used.

2 Algorithm Description

The algorithm this thesis is based on is summarized in the subsection 2.1, called *original* below.

For implementation in Gazebo, some modifications had to be made. These are described in subsection 2.2. The modified algorithm is referred to as *Gazebo-adjusted algorithm*.

In the subsection 2.3, a more explicit way of computing the covariance matrix used in both *original* and *Gazebo-adjusted* algorithms is described. This will be referred to as *the explicit algorithm*.

Subsection 2.4 shows a simple algorithm using a mixture of target position error and "intersection rule error" as the control input, thereafter called *the mixing algorithm*. (The *intersection rule* is explained in 2.1.6.)

2.1 The Original Algorithm by F. Poiesi and A. Cavallaro

2.1.1 Formation Shape

The algorithm works with any number of UAVs in formation greater than two members. For an illustration, let us assume there are three UAVs in a formation (also *observing UAVs*). Let us denote them *uav1*, *uav2*, and *uav3*. In the original algorithm the target (also *moving object*) is in a constant altitude. In this case, it can be another UAV. We can label it *uav4* or *target*.

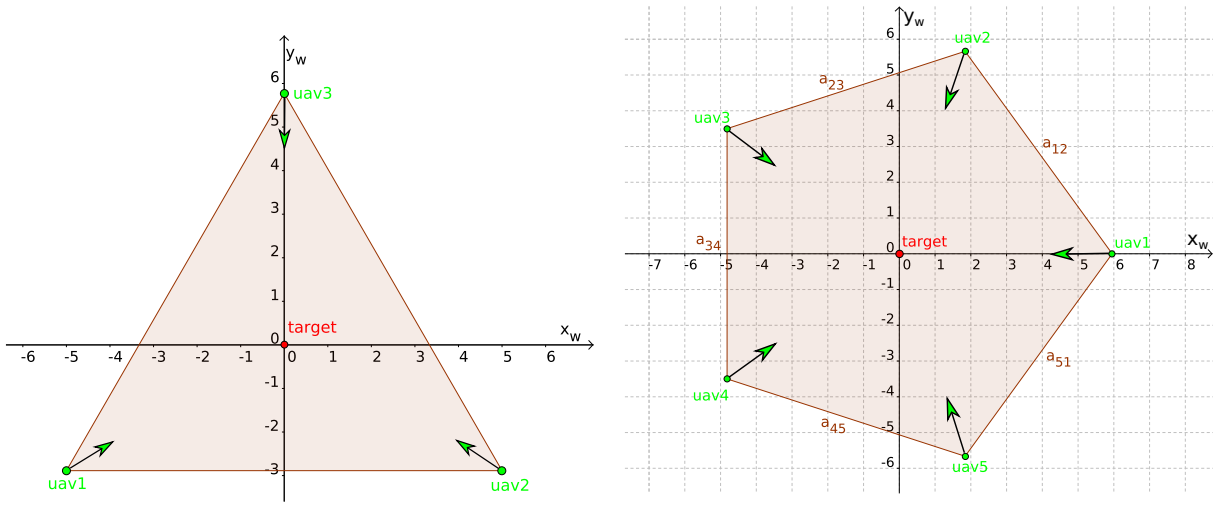
The initial state of the formation is an equilateral triangle with lengths of its sides a. The target is in the centre, in a different altitude. Each observing UAV is pointed towards the centre.

It is not necessary to use an equilateral triangle. Generally, there could be different default distances between UAVs, e.g. a_{12}, a_{23}, and a_{31}. Nevertheless, it is suggested that with this shape, the area covered by the fields of view of individual cameras is maximal.

Analogically, with higher number of UAVs, a regular polygon would be used.

An example of initial state seen from above is shown in figure 1a. The symbols x_w, y_w denote x- and y-axis of the world coordinate system. The initial state of 5-UAVs formation is in figure 1b.

A side view of the formation is shown in figure 2.



(a) Initial formation shape and position, $a_{12} = a_{23} = a_{31} = a = 10$ m

(b) Initial formation shape for 5 UAVs and their positions, $a_{12} = a_{23} = a_{34} = a_{45} = a_{51} = a = 7$ m

Figure 1: Possible initial states

2.1.2 General Notation

There are N UAVs in the formation, $i \in \mathbb{N}$ will be used for indexing of the UAVs. In this case, $N = 3$ and $i \in \{1, 2, 3\}$.

One UAV is allowed to communicate only with two adjacent UAVs. These UAV will be called *neighbours*, i.e. every UAV has two neighbours. For example, in case of the three-UAVs formation, *uav1* has neighbours *uav3* and *uav2*. The *uav2* has neighbours *uav1* and *uav3*, etc.

Unlike in [1], most variables will be named in a such way that it is easy to implement from the point of view of one UAV of the formation. Initial (default) distances of neighbouring UAVs are therefore $d_{L_i,init}, d_{R_i,init}$ – initial distance from a UAV to neighbour seen by i-th UAV on the left side, and the right side. In the case of the formaton in figure 1b, for $uav2$: $d_{L,init} = a_{12,init}, d_{R,init} = a_{23,init}$.

Additionally, discrete time $k \in \mathbb{N}_0$ will be used instead of continuous time.

All variables define the position in the UAV's coordinate frame unless the letter W is in the superscript, denoting the world coordinate system. Analogically, the letter r in the superscript denotes a frame with its origin in the respective UAV but with axis directions coincident with the world frame.

A position of the target in time step k in the world coordinate system is $\boldsymbol{x}_t[k] \in \mathbb{R}^3$. Similarly, the position of an observing UAV i in time step k in the world coordinate

system is $\boldsymbol{x}_i^W[k] \in \mathbb{R}^3$.

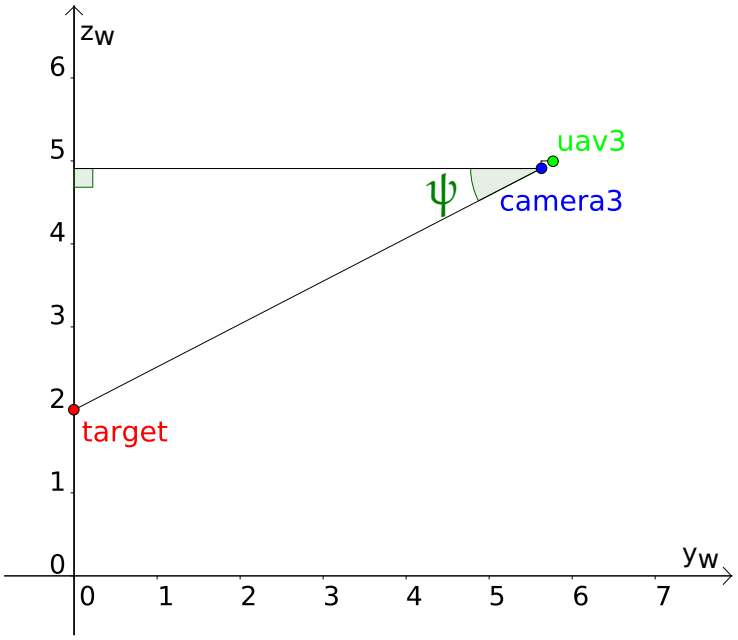


Figure 2: Side view of initial shape (one of the UAVs and the target is shown), with formation altitude 5 m, target altitude 2 m

2.1.3 Individual Error Processing

Each observing UAV has a static camera. Therefore, translation and rotation matrices defining its position with respect to the UAV coordinate system $\mathbb{T}_{camera}^{UAV}$ and $\mathbb{R}_{camera}^{UAV}$ are known. The rotation can be also defined by roll-pitch-yaw angles. We will denote the angle defining how is the forward tilt of the camera as ψ, as in seen in figure 2.

We also know the angle of view of each camera, the focal length, ..., the distortion coefficients and the camera projection matrix.

The most important property used for calculation of the covariance matrix (explained below) is the resolution of the camera image. An image plane $\left[-\frac{W}{2}, \frac{W}{2}\right] \times \left[-\frac{H}{2}, \frac{H}{2}\right]$, where W stands for width, and H stands for height, is assumed. I.e., the centre of the image has coordinates $[0, 0]$. Both in the simulator and with real cameras, the resolution $W = 1280$ px and $H = 720$ px is used (cameras are described in section 4.3)

Coordinates of the target *in the camera image* of i-th UAV are denoted as $\tilde{\boldsymbol{x}}_{t,i}[k] \in \mathbb{R}^2$.

Since the coordinates of the target in the camera plane and altitudes of both the target and observing UAV are known, it is possible to steer a UAV to follow it. As the camera

centre has the coordinates $\begin{bmatrix}0 & 0\end{bmatrix}^{\mathrm{T}}$, we can call $\tilde{\boldsymbol{x}}_{t,i}[k]$ the *error vector*. The first part of the algorithm is based on two simple ideas:

1. the bigger the deviation vector is, the faster observing UAV should move,
2. the deserved motion of the observing UAV depends on the direction of the error vector. The observing UAV should follow the error of the target from its ideal relative positon to the UAV.

In order to implement the first idea, a mapping $M : \mathbb{R}^2 \to \mathbb{R}$ is defined for calculating a magnitude. Let us define a covariance matrix $\Sigma_m \in \mathbb{R}^{2\times 2}$ which can be used to regulate the trend of M. The magnitude for i-th UAV is:

$$m_i\left[k\right] = M\left(\tilde{\boldsymbol{x}}_{t,i}\left[k\right]\right) = 1 - \exp\left(-\frac{1}{2}\cdot\tilde{\boldsymbol{x}}_{t,i}[k]^{\mathrm{T}}\cdot\Sigma_m^{-1}\cdot\tilde{\boldsymbol{x}}_{t,i}[k]\right). \tag{1}$$

An example of the mapping M values for the image resolution 1280 px × 720 px and the covariance matrix shown in equation (2) is plotted in figure 3. Notice that the trend of M in y-axis of the image is sharper to reflect the camera inclination. This is ensured by the covariance matrix:

$$\Sigma_m = \begin{bmatrix}80000 & 0\\ 0 & 24000\end{bmatrix} \tag{2}$$

To implement the second idea, formulas (3) and (4) are used:

$$a_{x,i}[k] = \operatorname{sgn}\left(\tilde{x}_{t,i}\left[k\right]\right)\cdot m_i\left[k\right], \tag{3}$$

$$a_{y,i}[k] = \operatorname{sgn}\left(\tilde{y}_{t,i}\left[k\right]\right)\cdot m_i\left[k\right]. \tag{4}$$

It is assumed here, that the camera heading is aligned with heading of the UAV. In [1], more general formulas are considered:

$$a_{x,i}[k] = \operatorname{sgn}\left(\tilde{\boldsymbol{x}}_{t,i}\left[k\right]\cdot\boldsymbol{e}_{x,i}\right)\cdot m_i\left[k\right], \tag{5}$$

$$a_{y,i}[k] = \operatorname{sgn}\left(\tilde{\boldsymbol{x}}_{t,i}\left[k\right]\cdot\boldsymbol{e}_{y,i}\right)\cdot m_i\left[k\right], \tag{6}$$

where $\boldsymbol{e}_{x,i}$, $\boldsymbol{e}_{y,i}$ are unite vectors rotated the same way as the camera frame to the UAV frame is: $\boldsymbol{e}_{x,i} = \mathbb{R}_{c,i}\cdot[1,0,0]\mathrm{T}$, $\boldsymbol{e}_{y,i} = \mathbb{R}_{c,i}\cdot[0,1,0]\mathrm{T}$.

This leads to a bounded set of possible vectors $\boldsymbol{a}_i[k]$. Firstly, the length is in $\langle 0,1\rangle$. Secondly, there are only 8 possible directions. In view of the UAV: forwards, backwards, to the right and to the left, and the diagonal variations thereof. As follows from the signum function definition, the former four are rarely used (only in case of $\tilde{x}_{t,i} = 0$ or $\tilde{y}_{t,i} = 0$).

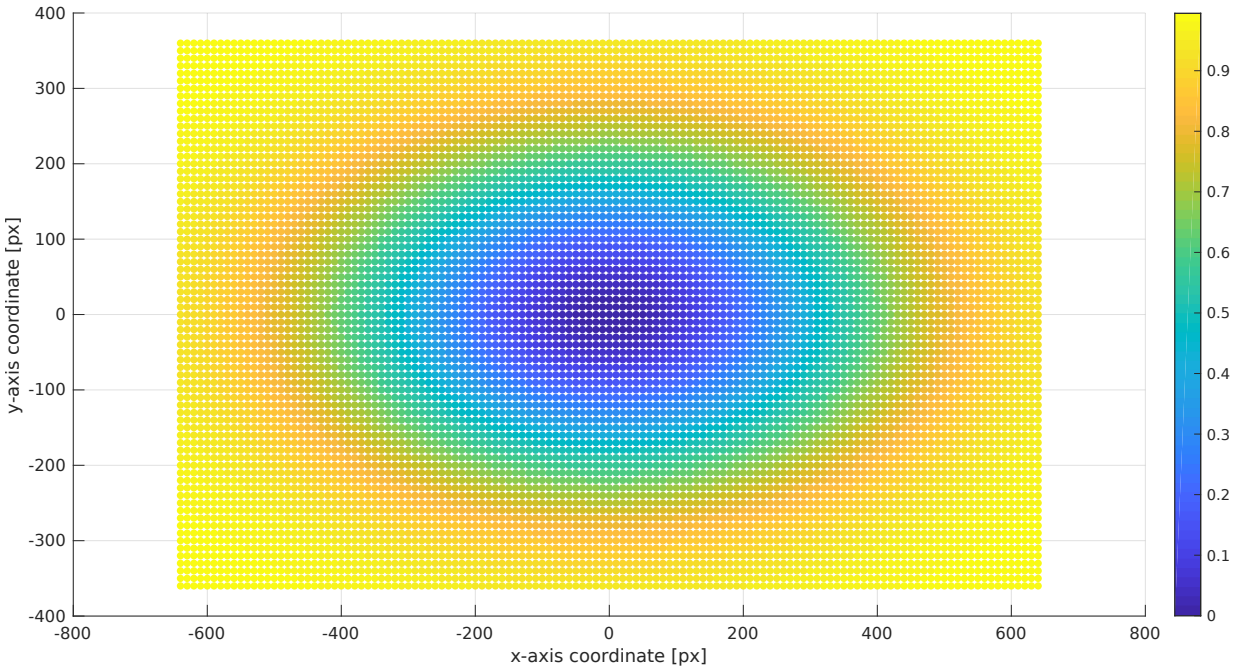


Figure 3: Magnitude distribution

2.1.4 Individual Velocity Setting

First of all the UAV candidate velocity $\boldsymbol{v}_i'$ is computed.

At this point, its useful to transform the thrust which was computed in the individual UAV reference frame to the world frame. Even without the RTK, the UAV has enough sensors to obtain its rotation. Therefore we can use the yaw angle ω (the rotation of the UAV around its z-axis). Other angles defining rotation are insignificant here.

$$\boldsymbol{a}_i^W[k] = \begin{bmatrix} cos(\omega) & -sin(\omega) \\ sin(\omega) & cos(\omega) \end{bmatrix} \cdot \boldsymbol{a}_i[k] = \begin{bmatrix} cos(\omega) & -sin(\omega) \\ sin(\omega) & cos(\omega) \end{bmatrix} \cdot \begin{bmatrix} a_{x,i}[k] \\ a_{y,i}[k] \end{bmatrix} \tag{7}$$

In order to decide when to stop the thrust action, time derivative of thrust is estimated as:

$$\dot{a}_{x,i}^W[k] = \left|a_{x,i}^W(k)\right| - \left|a_{x,i}^W(k-\tau_a)\right|, \tag{8}$$

$$\dot{a}_{y,i}^W[k] = \left|a_{y,i}^W(k)\right| - \left|a_{y,i}^W(k-\tau_a)\right|, \tag{9}$$

where τ_a is a small multiplication constant, e.g. 20, which is used to regulate the reaction speed. Larger values of τ_a may cause oscillations, while smaller values may lead to the loss of a fast-moving target.

The UAV candidate velocity in the world frame is then:

$$v'^{W}_{x,i}[k] = \begin{cases} \alpha \cdot a^{W}_{x,i}[k] + v^{W}_{x,i}[k-1], & \text{if } \dot{a}^{W}_{x,i}[k] > 0, \\ v'^{W}_{x,i} = v^{W}_{x,i}[k-1], & \text{otherwise;} \end{cases} \tag{10}$$

$$v'^{W}_{y,i}[k] = \begin{cases} \alpha \cdot a^{W}_{y,i}[k] + v^{W}_{y,i}[k-1], & \text{if } \dot{a}^{W}_{y,i}[k] > 0, \\ v^{W}_{y,i}[k-1], & \text{otherwise.} \end{cases} \tag{11}$$

The constant α affects the reaction of the UAV to the target dynamics. The bigger the α, the more aggressive the behaviour is.

2.1.5 Formation Velocity Agreement

If the target is visible to all the neighbours of a UAV, the desired/predicted velocity is equal to the arithmetic mean of the predicted velocities of its two neighbours:

$$\boldsymbol{v}^{W}_{i}[k] = \frac{1}{2}\left(\boldsymbol{v}'^{W}_{L}[k] + \boldsymbol{v}'^{W}_{R}[k]\right). \tag{12}$$

If the target is not detected by one of the neighbours, its predicted velocity is not taken into account:

$$\boldsymbol{v}^{W}_{i}[k] = \boldsymbol{v}'^{W}_{L}[k] \qquad \text{or} \qquad \boldsymbol{v}^{W}_{i}[k] = \boldsymbol{v}'^{W}_{R}[k], \quad \text{respectively.} \tag{13}$$

What is not explicitly mentioned in [1], is that if the target is detected by the corresponding UAV only, its desired velocity is used. (In a case of $N = 3$, not using this rule would result in a strong deceleration of the whole formation.)

In the remaining case (target is not detected either by any of the neighbours nor by the UAV itself), and in the case of a bad formation shape (defined by expression 19), a UAV is controlled according to the subsection 2.1.6 below.

If the velocity $\boldsymbol{v}^{W}_{i}[k]$ is known, then a desired trajectory candidate point is computed:

$$\boldsymbol{x}^{W,c}_{g,i}[k] = \boldsymbol{x}^{W}_{i}[k-1] + v^{W}_{i}[k] \cdot \Delta_k, \tag{14}$$

where Δ_k is the sampling time. Whether it will be used is decided in later steps.

2.1.6 Formation Shape Preservation

In the remaining cases, another desired position is used. First of all, an ideal position of the UAV is calculated as the intersection of two circles (see figure 4). Radii of the circles are the initial distances of *uav1* from its neighbours, $d_{L_1,init}$ for k_3 and $d_{R_1,init}$ for k_2. Centres are in the predicted positions of the neighbours. From these two intersection points, the closer one is selected, denoted $\boldsymbol{p}_i^{W,c}[k]$. This is the new desired position for the UAV, $\boldsymbol{x}_{g,i}[k]$.

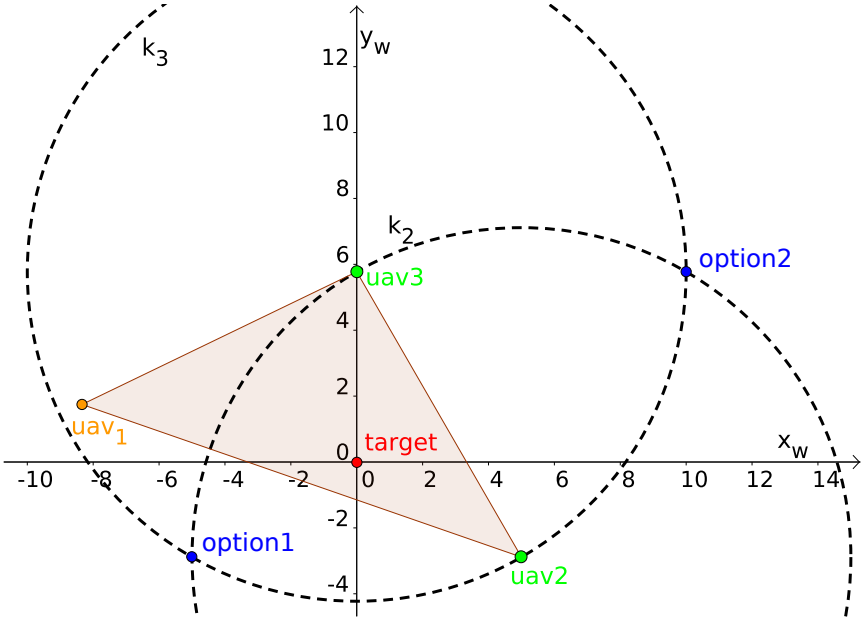


Figure 4: Intersection rule

This can be expressed as:

$$\begin{aligned} \boldsymbol{p}_i^{W,c} = \arg\min_x \left\{ \left\| \boldsymbol{x} - \boldsymbol{x}_{g,i}^{W,c}[k] \right\| \right\}, \\ x \in \Gamma(\boldsymbol{x}_{g,L_i}^{W,c}, d_{L_i,init}) \ \cap \ \Gamma(\boldsymbol{x}_{g,R_i}^{W,c}, d_{R_i,init}), \end{aligned} \tag{15}$$

where $\Gamma(\boldsymbol{c}, r)$ is a circle of radius r with the centre in $\boldsymbol{c}$.

After computing the desired velocity, formation shape is verified. To do this, each UAV computes ζ_L and ζ_R (see figure 5):

$$\zeta_{L_i}[k] = \left\| \boldsymbol{x}_{g,i}^{W,c}[k] - \boldsymbol{x}_{g,L_i}^{W,c}[k] \right\| - d_{L_i,init}, \tag{16}$$

$$\zeta_{R_i}[k] = \left\| \boldsymbol{x}_{g,i}^{W,c}[k] - \boldsymbol{x}_{g,R_i}^{W,c}[k] \right\| - d_{R_i,init}. \tag{17}$$

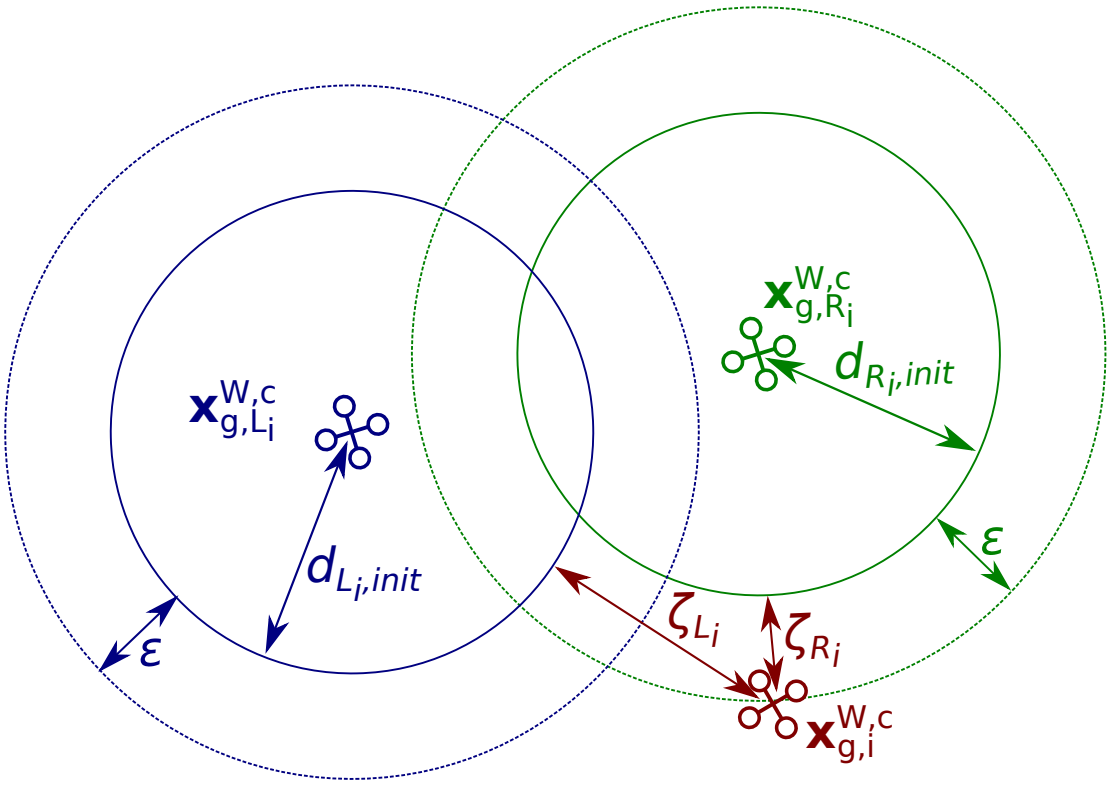


Figure 5: Formation shape varification

The "good formation shape" criterion is defined as

$$\zeta_{L_i} < \varepsilon \wedge \zeta_{R_i} < \varepsilon, \tag{18}$$

where ε is the distance threshold in metres. If this condition is valid, it is considered that the UAV is in a good position (the fromation has a good shape from point of view of this UAV) and the candidate trajectory point is used. Otherwise, the UAV is sent to the closer of the two circle intersections. In other words:

$$\boldsymbol{x}^{W}_{g,i}[k] = \begin{cases} \boldsymbol{x}^{W,c}_{g,i}[k], & \text{if } \zeta_{L_i} < \varepsilon \wedge \zeta_{R_i} < \varepsilon, \\ \boldsymbol{p}^{W,c}_{i}[k], & \text{otherwise.} \end{cases} \tag{19}$$

2.2 Differences Necessary for Our Implementation

2.2.1 Minor Changes

The changes not influencing the control mechanism were already included in the previous section:

- simpler equations (3) and (4) instead of (5) and (6) are used,
- to use the output as an input for MRS movement controller, $\boldsymbol{a}_i[k]$ must be rotated to the world frame, leading to $\boldsymbol{a}_i^W[k]$. Consequently, $\dot{\boldsymbol{a}}_i^W[k]$ and $\boldsymbol{v}_i'^W[k]$ are then used instead of $\dot{\boldsymbol{a}}_i[k]$ and $\boldsymbol{v}_i'[k]$.

2.2.2 Determining the Rotation of the UAV

What is not explicitly described in [1], is the desired heading of the UAV.

In the program, the yaw ω (about the z-axis of the UAV frame) is always set so that the UAV faces the centre of the connecting line of the neighbours $\boldsymbol{S}_i = [S_{x,i}, S_{y,i}]^{\mathrm{T}}$:

$$S_{x,i} = \frac{x_{g,L_i}^c + x_{g,R_i}^c}{2}, \tag{20}$$

$$S_{y,i} = \frac{y_{g,L_i}^c + y_{g,R_i}^c}{2}, \tag{21}$$

$$\psi_i = \mathrm{atan}_2(S_{y,i}, S_{x,i}), \tag{22}$$

where $\boldsymbol{x}_{g,L_i}^c$, $\boldsymbol{x}_{g,R_i}^c$ are the candidate positions of the neighbours in its frame.

2.2.3 UAV Control Specifics

As stated in [11], a "hierarchical control approach is common for quadrotors, with the lowest control level being control of the rotor rotational speed. The next level is control of vehicle attitude, and the top level is control of a quadcopter position…". The motion controller of the UAV is not an exception, and using the topmost level of the control is preferred (e.g. the collision avoidance is already implemented in this level). It accepts as its input one of following:

1. a single setpoint consisting of the x, y, and z coordinates and optionally also ψ in the world coordinate system,
2. a sequence (trajectory) of absolute setpoints defined above, distributed with spacing of 0.2 seconds,
3. a single setpoint of the same format, relative to the current position of UAV.

The second option makes it possible to set the velocity by using regularly distributed points. Problems with the collision avoidance mechanism occured due to the trajectories

being long and occasionally stepping in the safety zones of the other UAVs. As the control mechanism is based on relative localization, the third option was chosen in the end.

Instead of $\boldsymbol{x}_{g,i}^{W,c}[k]$ from equation (14) or $\boldsymbol{p}_i^{W,c}[k]$ defined in (15), forms relative to the UAV position must be used:

$$\boldsymbol{x}_{g,i}^{r,c}[k] = v_i^W[k] \cdot \Delta_k, \tag{23}$$

$$\begin{aligned} \boldsymbol{p}_i^c = &\arg\min_x \left\{ \left\| \boldsymbol{x} - \boldsymbol{x}_{g,i}^c[k] \right\| \right\}, \\ &x \in \Gamma(\boldsymbol{x}_{g,L_i}^c, d_{L_i,init}) \ \cap \ \Gamma(\boldsymbol{x}_{g,R_i}^c, d_{R_i,init}), \end{aligned} \tag{24}$$

$$\boldsymbol{p}_i^{r,c}[k] = \begin{bmatrix} cos(\omega) & -sin(\omega) \\ sin(\omega) & cos(\omega) \end{bmatrix} \cdot \boldsymbol{p}_i^c[k], \tag{25}$$

$$\boldsymbol{x}_{g,i}^r[k] = \begin{cases} \boldsymbol{x}_{g,i}^{r,c}[k], & \text{if } \zeta_{L_i} < \varepsilon \wedge \zeta_{R_i} < \varepsilon, \\ \boldsymbol{p}_i^{r,c}[k], & \text{otherwise.} \end{cases} \tag{26}$$

The sampling time used is small compared to the one used by the high-level motion controller. For this reason, setting very small setpoints computed in (23) results in no movement.

The further setpoint is sent to the MPC, the faster the UAV will move. To amplify the output, we will be using the constant β:

$$\boldsymbol{x'}_{g,i}^r[k] = \beta \cdot \boldsymbol{x}_{g,i}^r[k]. \tag{27}$$

2.2.4 Omitting of the Position Prediction

Again, because of the processing rate of the program (25 Hz), it is admissible not to predict the position of the neighbours. This is because the change in the output of the control algorithm will not have a significant effect.

In the algorithm, only the current positions of both the UAV and the neighbours are used in (15)/(24):

$$\begin{aligned} \boldsymbol{x}_{g,i} &\text{ instead of } \boldsymbol{x}^c_{g,i}, \\ \boldsymbol{x}_{g,L_i} &\text{ instead of } \boldsymbol{x}^c_{g,L_i}, \\ \boldsymbol{x}_{g,R_i} &\text{ instead of } \boldsymbol{x}^c_{g,R_i}. \end{aligned} \tag{28}$$

Additionally, equations (16), (17) are used in following form:

$$\zeta_{L_i}[k] = \|\boldsymbol{x}_{g,i}[k] - \boldsymbol{x}_{g,L_i}[k]\| - d_{L_i,init}, \tag{29}$$

$$\zeta_{R_i}[k] = \|\boldsymbol{x}_{g,i}[k] - \boldsymbol{x}_{g,R_i}[k]\| - d_{R_i,init}. \tag{30}$$

2.2.5 Suppression of Oscillations

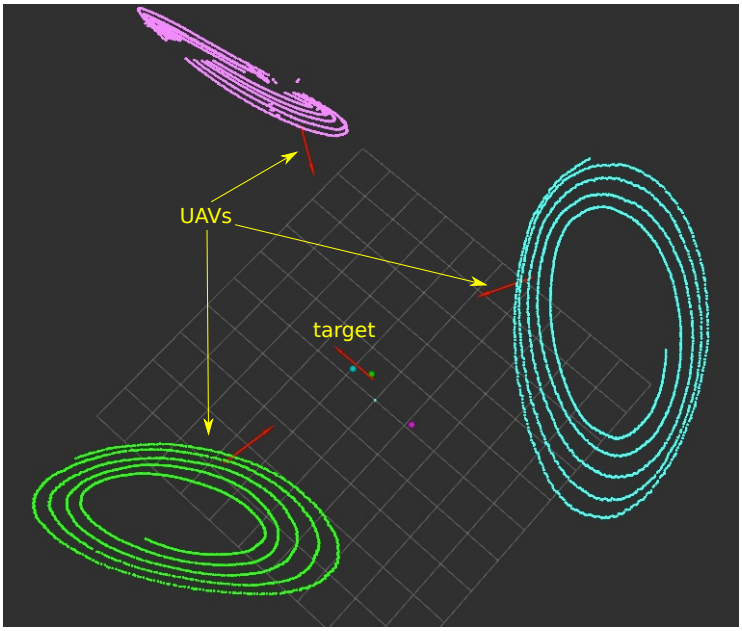


Figure 6: An oscillating formation of three UAVs

In simulations, the oscillations of both the individual UAVs and the formation as a whole (see figure 6, a screenshot from RViz[2]) were observed. In the figure, the red arrows represent odometry data. The turquoise, green and pink spirals consist of the respective setpoints. Therefore, some tools to suppress them were added to the program.

[2]RViz is a ROS package for topics visualization, see `http://wiki.ros.org/rviz`.

The main tool is an exponential filter. The output is modified as follows:

$$\boldsymbol{x}''^{r}_{g,i}[k] = q \cdot \boldsymbol{x}'^{r}_{g,i}[k] + (1-q) \cdot \boldsymbol{x}''^{r}_{g,i}[k-1], \tag{31}$$

where $q \in \langle 0, 1\rangle$ is a constant.

The reference change of the yaw angle for the MPC is descreased by multiplying a γ constant ($\gamma \in (0, 1)$) because of the same reason.

As an anti-windup filter, the predicted velocity is limited. Mathematically:

$$v''^{W}_{x,i}[k] = \begin{cases} v'^{W}_{x,i}[k], & \text{if } v'^{W}_{x,i}[k] < v_{max}, \\ v_{max}, & \text{otherwise.} \end{cases} \tag{32}$$

In pogram, there is additionally an option to apply a deadband:

$$\boldsymbol{x}'^{r}_{g,i}[k] = \begin{cases} [0,0]^T, & \text{if } |m_i[k]| < m_{max}, \\ \boldsymbol{x}^{r}_{g,i}[k], & \text{otherwise.} \end{cases} \tag{33}$$

In order not to switch between target following and approaching the intersection point too often, a hysteresis condition was added. The UAV follows the intersection point as long as

- bad formation shape was detected,
- bad formation shape has been detected up to `distanceTimeout` seconds ago.

2.2.6 Formation Shape Verification Condition

For small formations, the formation shape verification condition (see equation (18)) does not work properly. It allows the UAVs to go very close to their neighbours. The difference is in the ideal distance $d_{n,ideal}$ of the neighbours of the UAV:

- for $N = 3$, $d_{n,ideal} = a$,
- for $N = 4$, $d_{n,ideal} = a\sqrt{2}$,
- for $N \to \infty$, $d_{n,ideal} \to 2a$.

There is always a "corridor" between the neighbours, where the UAV can be and where it is still considered a good formation shape. Whereas in the case of $N \to \infty$, it is only 2ε broad, in case of $N = 3$, it is wider more than the ideal distance $d_{n,ideal}$ itself.

Another rules is therefore used. Formation shape is considered to be wrong, if and only if:

$$\left| \left\| \boldsymbol{x}_{g,i}^{W,c}[k] - \boldsymbol{x}_{g,L_i}^{W,c}[k] \right\| - d_{L_i,init} \right| > \varepsilon \quad \vee \quad \left| \left\| \boldsymbol{x}_{g,i}^{W,c}[k] - \boldsymbol{x}_{g,R_i}^{W,c}[k] \right\| - d_{R_i,init} \right| > \varepsilon. \tag{34}$$

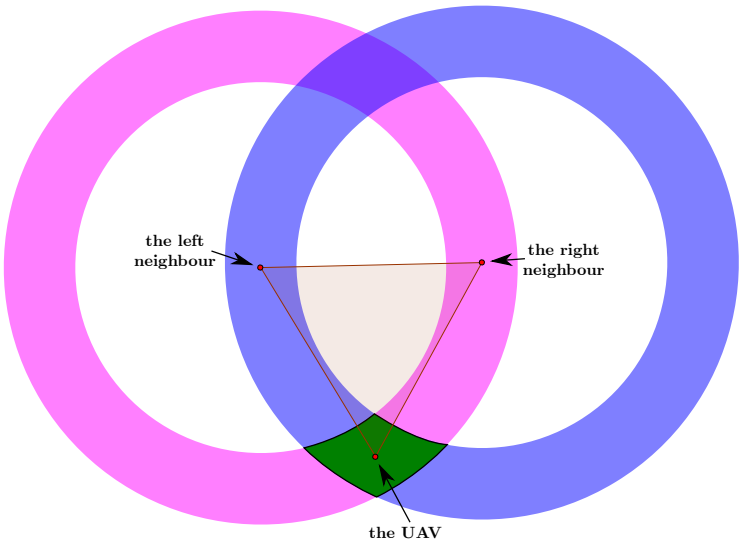


Figure 7: The new formation shape criterion

A graphical explanation is shown in figure 7. It shows a situation of the ideal formation shape for $N = 3$, $a = 6$ m, $\varepsilon = 1$ m. The green area represents all the positions of the UAV where 34 is false.

2.2.7 Behaviour in the Case of No Target Detection

In case of no target detection, the current velocity is sent to its neighbours.

Provided that the target is not detected, in cases of neighbours undetection (or simply when the intersection does not exist), the relevant UAV positions itself according to the only detected neighbour (an ideal position for the formation is computed beforehand), or stays in the current place.

The rules up to this point describe *the original algorithm.*

2.3 Exact Thrust Direction Computation

This subsection describes *the exact algorithm.*

2.3.1 Motivation

In the original algorithm, the control input is converted into the acceleration vector using the matrix Σ_m and changes of direction (equations (3)–(4) or (5)–(6), and (7)).

It does not deal with "temporary" UAV rotations in pitch and roll directions. "As the disturbance, the movement of the UAV has a big influence on the performance of the tracking system. These disturbances should be compensated to improve the tracking performance. [9]"

For example, if the target is detected in the upper part of camera image, the UAV will move forward. Because of the nature of UAV (e.g. quadro- or hexacopter) motion behaviour, in order to move forward, it must to lean forward. This leaning shifts the target in the camera image downwards a lot, even though the UAV might have moved forward only a short distance.

For another example, when the camera angle is changed, the covariance matrix should be changed to keep the algorithm working properly. This can be done either experimentaly, in a naive way, or we can prepare one covariance matrix and modify it with transformation matrices later which saves time.

We can compute the position of the target relative to the UAV (from camera matrix and current rotation matrix of the UAV) first, and use this value for computation of the acceleration.

2.3.2 Camera Simulation

In order to test the formation algorithms independently of computer vision, we use a helper ROS node. It takes as input the odometry data of an observing UAV and of the target in the world frame from relevant topics and computes the coordinates of the target in the camera image. From the target odometry, it uses only the position, represented as the vector $\boldsymbol{x}_t^W$. Homogeneous coordinates will be used:

$$\boldsymbol{x}_t^W = \begin{bmatrix} x_t^W \\ y_t^W \\ z_t^W \\ 1 \end{bmatrix}, \qquad \mathbb{T}_{UAV}^W = \begin{bmatrix} 1 & 0 & 0 & x_t^W \\ 0 & 1 & 0 & y_t^W \\ 0 & 0 & 1 & z_t^W \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{35}$$

where $\mathbb{T}_{UAV}^{W}$ is translation matrix of the UAV with respect to the world coordinate system. Analogically, $\mathbb{R}_{UAV}^{W}$ is a matrix defining rotation of the UAV with respect to the world frame. Matrices defining the pose of the camera to the UAV frame are denoted $\mathbb{T}_{camera}^{UAV}$, $\mathbb{R}_{camera}^{UAV}$. For example, these were used in experiments:

$$\mathbb{T}_{camera}^{UAV} = \begin{bmatrix} 1 & 0 & 0 & 0.211\text{ m} \\ 0 & 1 & 0 & 0.0\text{ m} \\ 0 & 0 & 1 & -0.05\text{ m} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbb{R}_{camera}^{UAV} = \begin{bmatrix} \cos(0.7\text{ rad}) & 0 & \sin(0.7\text{ rad}) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(0.7\text{ rad}) & 0 & \cos(0.7\text{ rad}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{36}$$

One of the coordinate systems for an image used in computer vision (CV) is that with origin in the upper-left corner, x-axis to the right and y-axis heading down. To represent this change, there are special matrices

$$\mathbb{R}_{x,CV}^{camera} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbb{R}_{z,CV}^{camera} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{37}$$

representing two simple rotations by right angles around x- and z-axis.

Knowing also a camera matrix $\mathbb{P}$ (see [14]), it is possible to compute the normalized coordinates $\boldsymbol{N}$ of the target:

$$\boldsymbol{N} = \mathbb{P} \cdot \left(\mathbb{R}_{x,CV}^{camera}\right)^{-1} \cdot \left(\mathbb{R}_{z,CV}^{camera}\right)^{-1} \cdot \left(\mathbb{R}_{camera}^{UAV}\right)^{-1} \cdot \left(\mathbb{T}_{camera}^{UAV}\right)^{-1} \cdot \left(\mathbb{R}_{UAV}^{W}\right)^{-1} \cdot \left(\mathbb{T}_{UAV}^{W}\right)^{-1} \cdot \boldsymbol{x}_t^W \tag{38}$$

Coordinates in pixels, denoted u, v, are then calculated ([15], also follows from [14]):

$$\begin{aligned} u &= \frac{N_1}{N_3}, \\ v &= \frac{N_2}{N_3}. \end{aligned} \tag{39}$$

2.3.3 Reprojection

We are looking for an intersection of a horizontal plane assuming that the target has constant altitude with a vector.

To express the task in language of the matrices introduced in the previous subsection, we can rewrite the equation 38 in a reverse order. As we want to know the relative position of target to the UAV, matrices $\mathbb{T}^W_{UAV}$ and $\mathbb{R}^W_{UAV}$ must be removed:

$$\boldsymbol{x}^r_t = \mathbb{T}^{UAV}_{camera} \cdot \mathbb{R}^{UAV}_{camera} \cdot \mathbb{R}^{camera}_{z,CV} \cdot \mathbb{R}^{camera}_{x,CV} \cdot \mathbb{P}^{-1} \cdot \boldsymbol{N}. \tag{40}$$

Rotating by the yaw angle, we get the point in the frame of the UAV

$$\boldsymbol{x}_t = \begin{bmatrix} \cos\omega & 0 & \sin\omega & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\omega & 0 & \cos\omega & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \boldsymbol{x}^r_t. \tag{41}$$

Using following notation:

$$\boldsymbol{x}_t = \mathbb{M} \cdot \boldsymbol{N}, \tag{42}$$

$$\boldsymbol{x}_t = \begin{bmatrix} x_t \\ y_t \\ z_t \\ 1 \end{bmatrix}, \quad \boldsymbol{N} = \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ 1 \end{bmatrix}, \quad \mathbb{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{43}$$

equations for the normalized coordinates (39), and knowing that z_t is a constant value, we can solve equation (42) as a system of 6 linear equations with 5 unknowns. The solution is:

$$N_3 = \frac{-z_t - m_{34}}{m_{31} \cdot u + m_{32} \cdot v + m_{33}}, \tag{44}$$

$$\begin{aligned} x_t &= m_{11} \cdot u \cdot N_3 + m_{12} \cdot v \cdot N_3 + m_{13} \cdot c + m_{14}, \\ y_t &= m_{21} \cdot u \cdot N_3 + m_{22} \cdot v \cdot N_3 + m_{23} \cdot c + m_{24}. \end{aligned} \tag{45}$$

2.3.4 Relative Position Usage

The relative target position $\boldsymbol{x}_t$ from subsection 2.3.3 is then used in the same way as the position in camera image $\tilde{\boldsymbol{x}}_t$ is. I.e., instead of (1), the following equation is used:

$$m_i\left[k\right] = M\left(\boldsymbol{x}_{t,i}\left[k\right]\right) = 1 - \exp\left(-\frac{1}{2} \cdot \boldsymbol{x}_{t,i}[k]^{\mathrm{T}} \cdot \Sigma_m^{-1} \cdot \boldsymbol{x}_{t,i}[k]\right). \tag{46}$$

(Specifically, the program uses $\boldsymbol{x}_{t,i}$ in millimeters.)

Another difference of this algorithm with respect to the original is how we determine the direction of $\boldsymbol{a}$. We rotate it by the angle of target error vector:

$$\nu = \mathrm{atan}_2(y_{t,i}, x_{t,i}), \tag{47}$$

$$\boldsymbol{a} = \begin{bmatrix} \cos(\nu) & -\sin(\nu) \\ \sin(\nu) & \cos(\nu) \end{bmatrix} \cdot \begin{bmatrix} m_i[k] \\ 0 \end{bmatrix} = \begin{bmatrix} m_i[k] \cdot \cos(\nu) \\ m_i[k] \cdot \sin(\nu) \end{bmatrix}. \tag{48}$$

2.4 Setpoint Mixing Algorithm

The algorithm described in this section is reffered as to *the mixing algorithm.*

2.4.1 Motivation

When the UAV controller according to the previous two algorithms detects that it is in a wrong relative position with respect to its neighbours, it completely switches from following the target to following a virtual point defined by the position of its neighbours, even though its neighbours can still be following the target, as opposed to keeping the formation shape. This point therefore may be delayed, the UAV follows a point which follows the neighbours, while it may take a long time to detect the target again for this UAV.

2.4.2 Mathematical Description

In order to mitigate this effect, the setpoint mixing algorithm takes the relative target position, the position of the closer intersection (both in x-y plane), and computes the weighted arithmetic mean. The result is multiplied by the constant β introduced in subsection 2.2.3:

$$\boldsymbol{x}_{g,i}[k] = \beta \cdot \frac{(1+w) \cdot \boldsymbol{x}_{t,i}^{c}\left[k\right] + (1-w) \cdot \boldsymbol{p}_i^{c}\left[k\right]}{2}, \tag{49}$$

$$\boldsymbol{x}_{g,i}^{W}[k] = \begin{bmatrix} cos(\omega) & -sin(\omega) \\ sin(\omega) & cos(\omega) \end{bmatrix} \cdot \boldsymbol{x}_{g,i}[k], \tag{50}$$

where $w \in \langle -1, 1 \rangle$ is a constant.

If one of the points can not be computed, the remaining point is used:

$$\boldsymbol{x}_{g,i}[k] = \begin{cases} \boldsymbol{x}_{t,i}^{c}[k], & \text{if it exists,} \\ \boldsymbol{p}_{i}^{c}[k], & \text{if it exists,} \\ [0,0]^{\mathrm{T}}, & \text{if neither one exists.} \end{cases} \tag{51}$$

2.5 Adjustment for a Ground Target

Up to this point, only a target flying in a constant altitude or a ground target moving on a flat ground were considered. Nevertheless, one of the aims of this thesis is to follow a ground target while the ground may not be perfectly flat.

One way to do so is to keep the formation in a horizontal plane of a constant altitude, measuring the distance from the ground and using this for correction of the target position estimate. As the MRS drones have different behaviour, another approach was chosen. The MRS drones are keeping a *constant height individually*, with respect to the ground under them.

This means the UAVs follow the surface. Flying above a slant surface results in a non-horizontal formation. We will assume a gradually changing surface, so that the surface of the ground covered by the formation can be always considered planar. It could also be assumed that a ground target does not change its height. The same applies as a restriction for a flying target, it also cannot change its height.

As a UAV does not know its absolute position, all the calculations will be done in its reference frame. Since it tries to keep a constant distance from the ground, this will be considered constant and known. To define a plane, three points are needed. For the purposes above, positions of its neighbours are already detected. Hence we have three points defining the plane of the formation (positions of the left and the right neighbour and the position of the UAV itself as origin).

Considering the positions of the neighbours as vectors lying in the plane, it is easy to compute the normal vector of the plane in view of i-th UAV:

$$\boldsymbol{n}_i[k] = [n_{1,i}[k], n_{2,i}[k], n_{3,i}[k]]^{\mathrm{T}} = \boldsymbol{x}_{R_i}[k] \times \boldsymbol{x}_{L_i}[k]. \tag{52}$$

As both the formation and the target are keeping the same distance from ground, the distance of the formation and target plane should also be constant and they should be parallel. In section 2.3, the relative position of the target was calculated, assuming the same planes of the same distance with horizontal inclination. We can use this position as an orientation

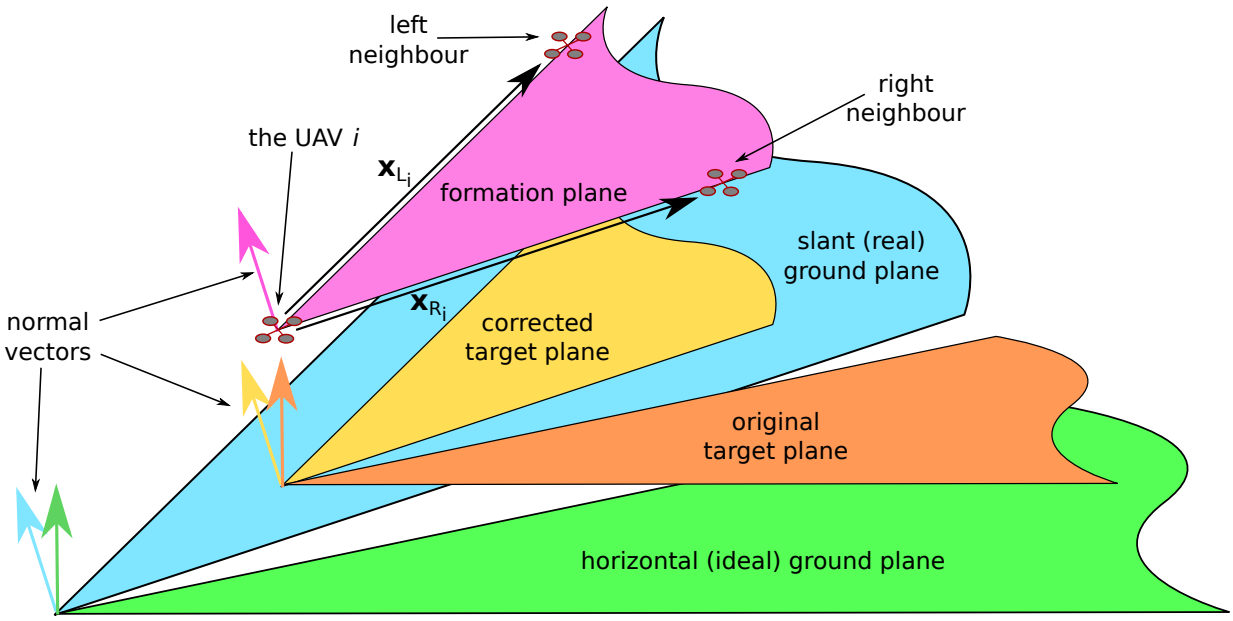


Figure 8: Illustration of the planes

vector and find the intersection of it with the slant target plane. All the planes are depicted in figure 8.

The equation of a plane is:

$$n_1 \cdot x_t + n_2 \cdot y_t + n_3 \cdot z_t + n_4 = 0, \tag{53}$$

where x_t, y_t, z_t are target coordinates in the UAV's frame. Constant n_4 has to be calulated — to make it an equation of the target plane, the coordinates x_t, y_t, z_t must be substituted with a point in the plane. (i in indices and the time step specification k is ommited here) This can be the point directly under the UAV, $[0, 0, -h_{init}]^{\mathrm{T}}$, which leads to

$$n_4 = h_{init} \cdot n_3. \tag{54}$$

If we express the line defined directional vector parametrically (see figure 9):

$$\begin{aligned} x &= t_1 \cdot p, \\ y &= t_2 \cdot p, \\ z &= t_3 \cdot p, \end{aligned} \tag{55}$$

where t_1, t_2, and t_3 are the original target coordinates computed in section 2.3, and we put these into the plane equation, the parameter p is expressed as:

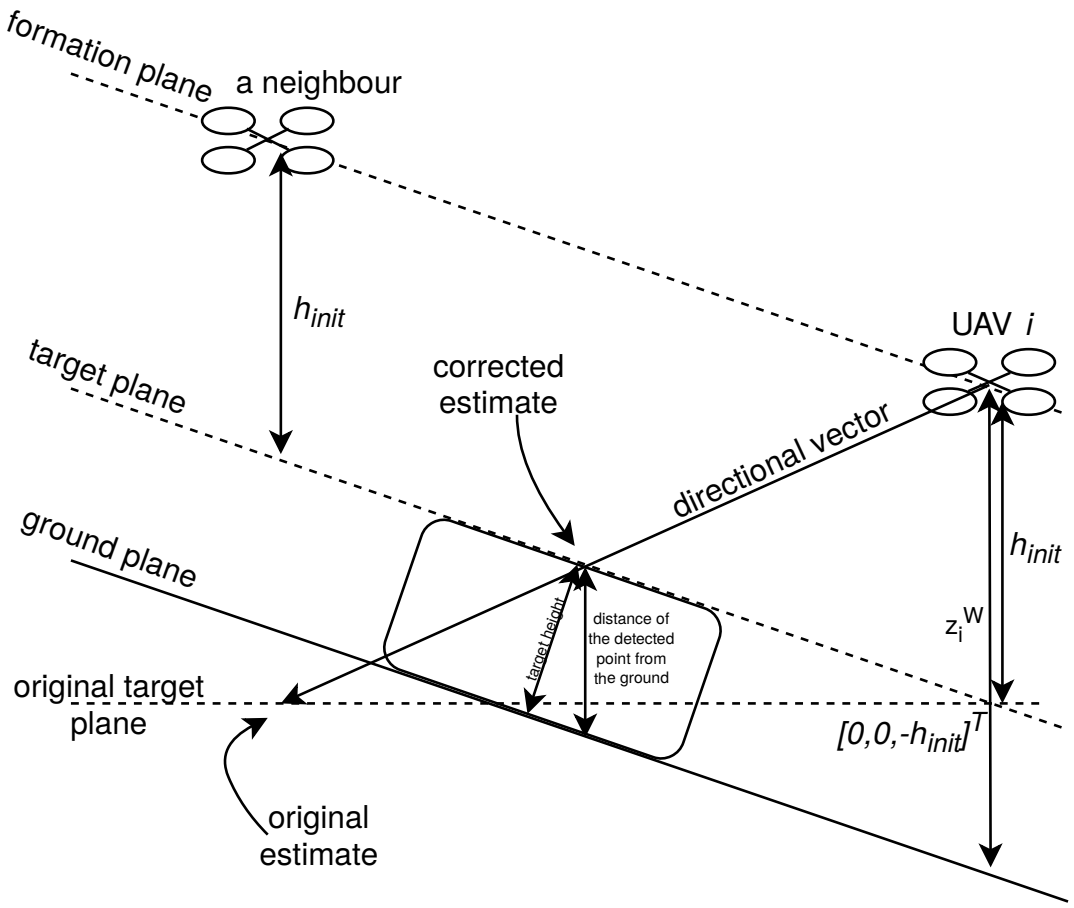


Figure 9: Slope profile

$$p = \frac{n_3 \cdot h_{init}}{-n_1 \cdot t_1 - n_2 \cdot t_2 + n_3 \cdot h_{init}}. \tag{56}$$

Knowing the more precise target coordinates, these can be used in *the explicit* algorithm or *the mixing* algorithm.

2.6 Implementation Details

A single ROS node which produces the desired trajectory of a corresponding UAV was programmed. Accoring to passed parameters, it runs certain algorithm.

For simulations with a UAV as the target, another node was programmed, described in subsection 2.3.2.

For other simulations, a ROS version of the Whycon system (WhyCon-ROS) is used for detection of the target as well as of the neighbours. As this version does not support differet patterns (different IDs), the node decides what detected object is the target, left neighbour, or the right neighbour. This is done in a naive way, based on areas of the image. The target is expected to be in the lower area of the image, the left neighbour in upper left area, and the right neighboour in the upper right area. The areas are adapted to the inclination of the UAV and if the detected patterns are not in the expected positions (eg. a neighbour is too close to the target), an error is reported.

Additionally, this detector has has problems when attempting to detect a certain number of patterns in an image where a lower number is visible. It it is the case, it takes a long time to produce an output. Therefore, there are multiple nodes run simultaneously, each set to detect a different number of patterns in the image. Close detections close to each other (there is 1 m distance threshold) are then grouped and averaged.

A version of Whycon able to distinguish among patterns was modified to produce ROS messages. However, this version produces a large number of false detections to the effect that multiple were assigned IDs to the same pattern for pattern sizes and distances used in simulations.

3 Simulation results

For testing of the algorithms, a simple trajectory for the followed UAV was chosen. It consists of regularly distributed points so that the UAV velocity is about 0.8 m/s most of the time. There is a 10-seconds-long pause in a turning point starting in $t = 40$ s. The complete trajectory length is 60 s.

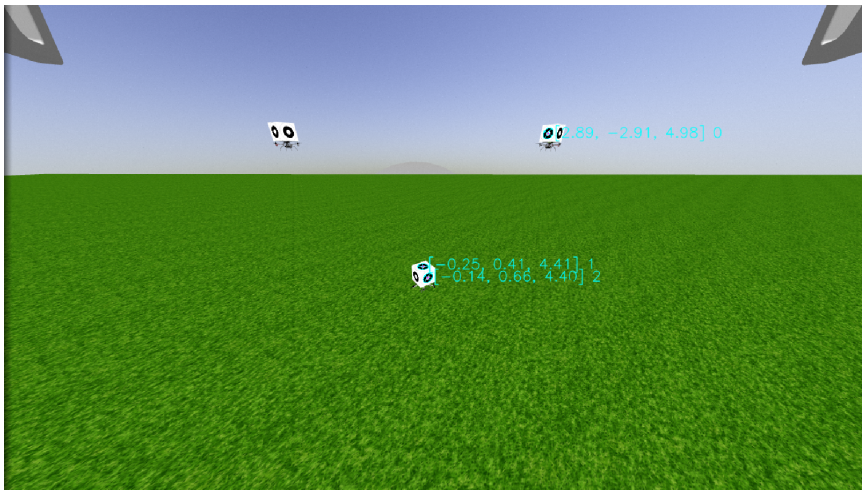


Figure 10: The relative localization of the neighbours

The configuration file for testing of the original algorithm can be seen in figure 34. The trajectory of the formation centre with the original algorithm is shown in the figure 11.

The algorithms were mostly tested with the computed target coordinates and neighbours localization from the odometry data in a planar environment: the results are shown in the figures 13, 36 (formations of 3 UAVs controlled by the *exact* algorithm), 37, 38, 39 (formations of 3 UAVs controlled by the *mixing* algorithm), 40 (a formation of 4 UAVs controlled by the *exact* algorithm), and 41 and 42 (4 UAVs, the *mixing* algorithm). For all of them, $a = 6$ was set. The basic configuration file is shown in the figure 35, the important distinctions are described in the corresponding captions.

Target estimation was tested in a world with hill surface and a desk with pattern simulating a UGV, see the figure 12. An experiment from this environment is shown in the figure 14. Additionally, the functionality of the localization of the neighbours using the Whycon system was verified in the simulations (see figure 10).

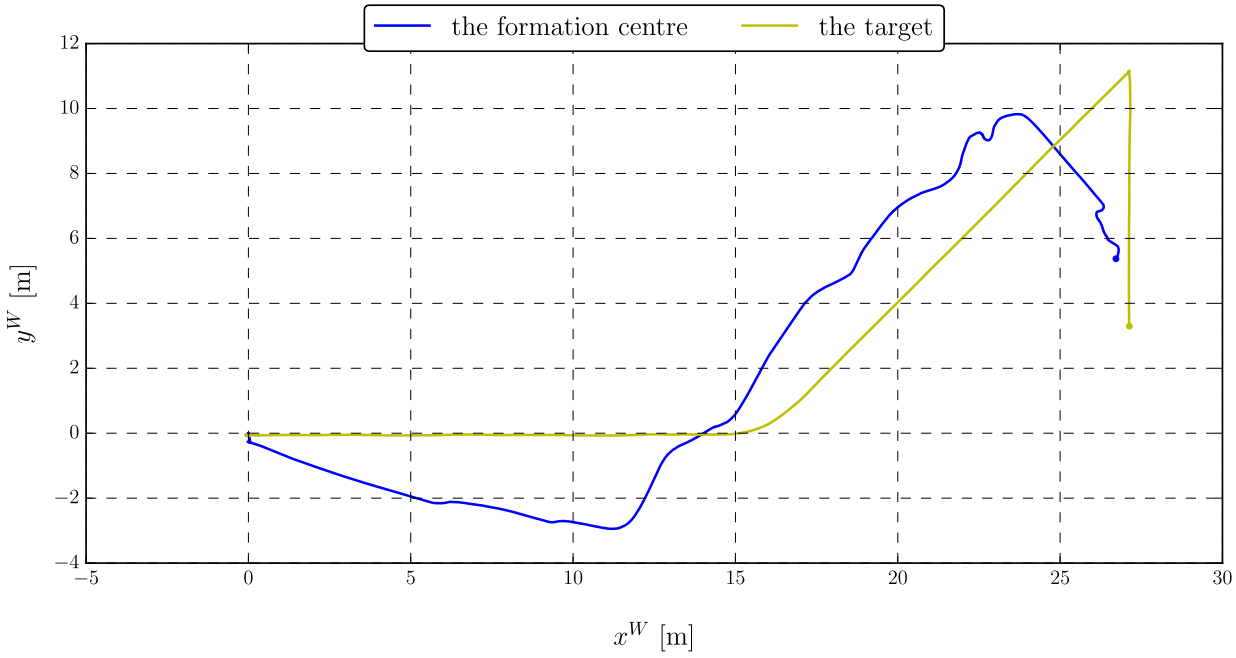


Figure 11: The trajectory of the formation centre with the original algorithm

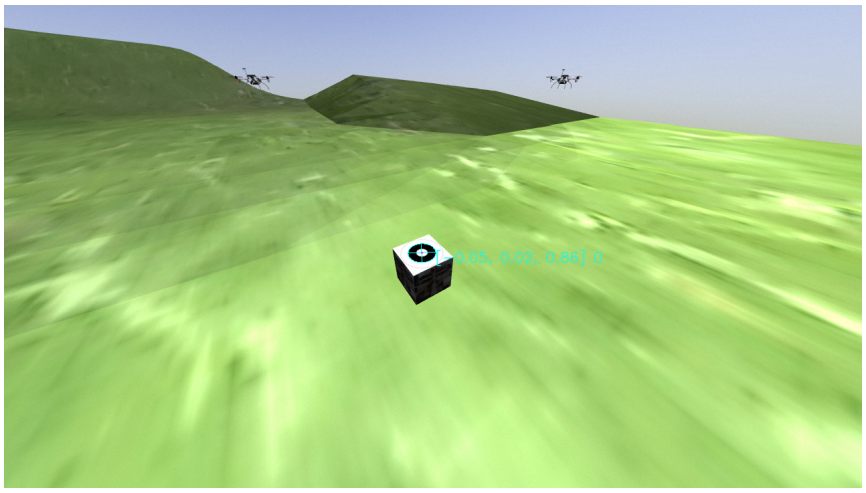
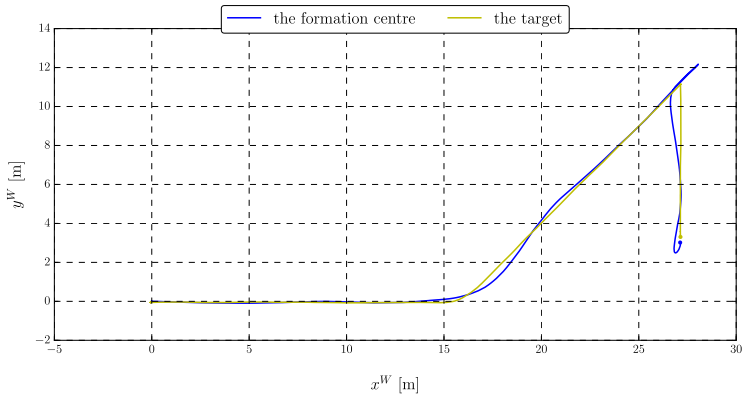
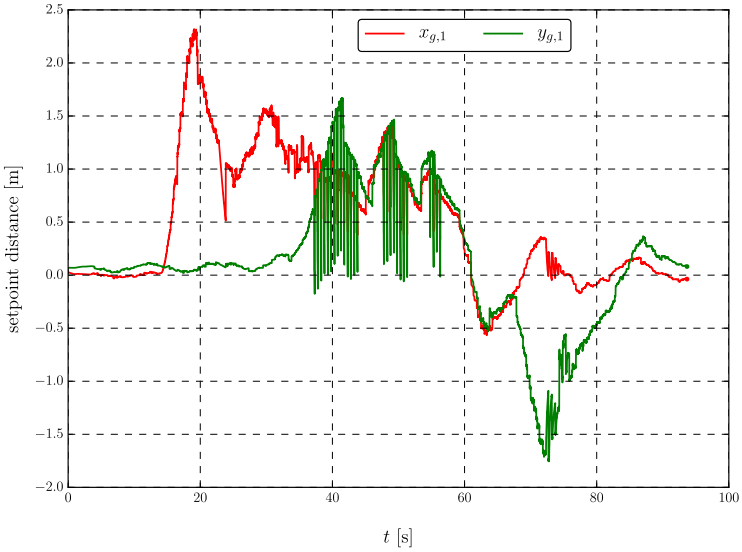


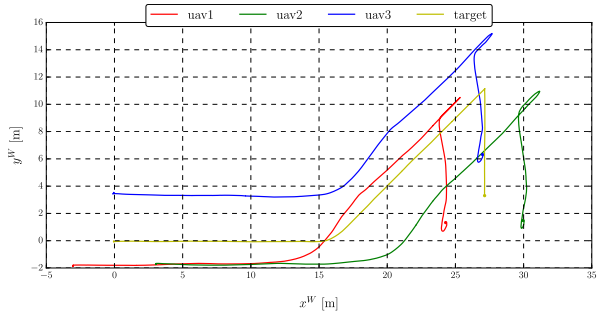
Figure 12: The testing environment



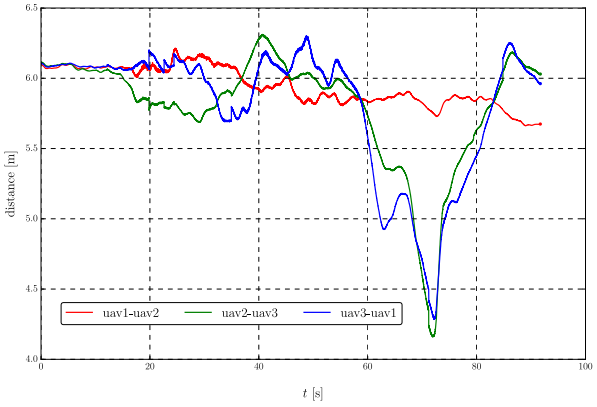
(a) The target and the formation centre trajectories



(b) The setpoints of *uav1*

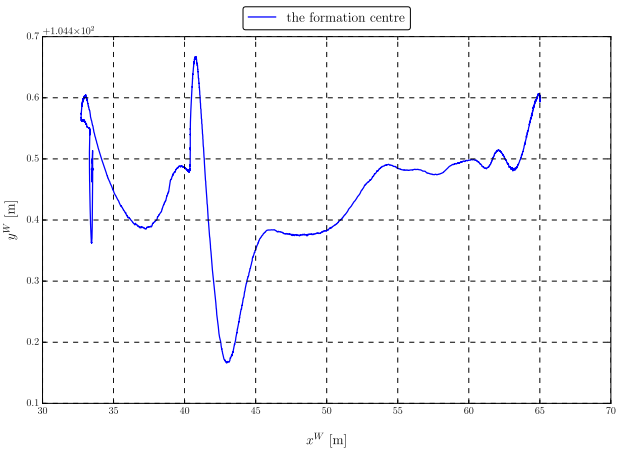


(c) Overall view (all trajectories)

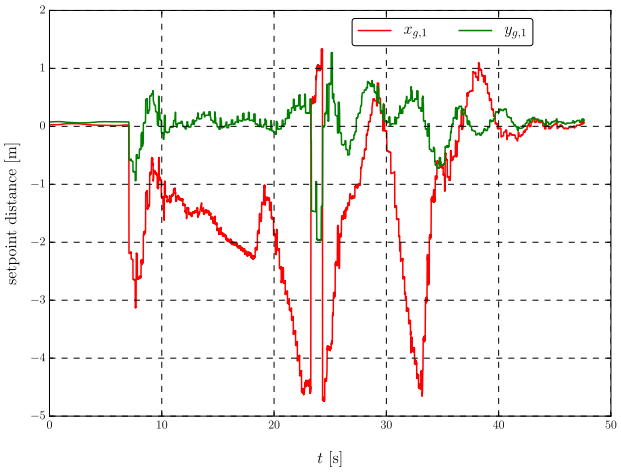


(d) The mutual distances of the neighbours

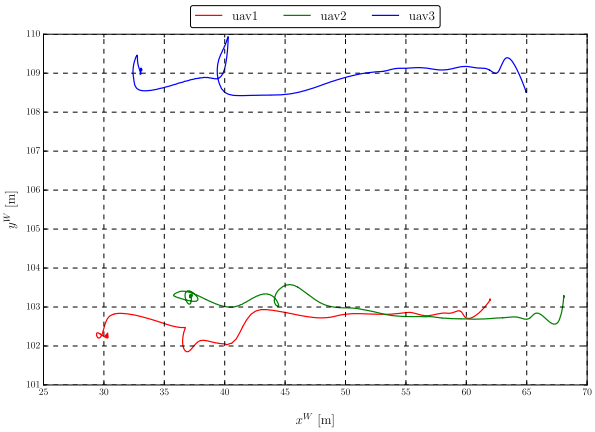
Figure 13: The experiment with 3 UAVs, *the exact* algorithm, $\alpha = 1.5$, $\beta = 1.0$



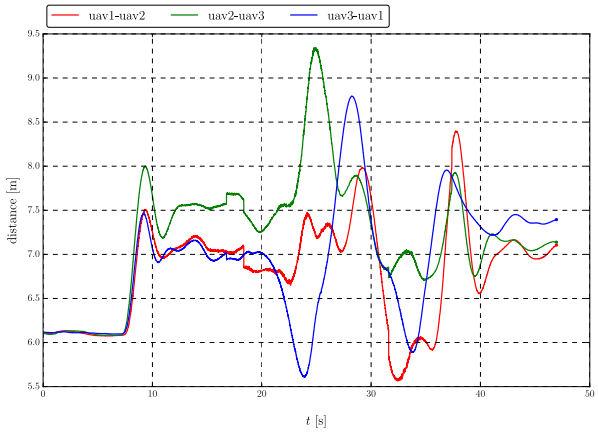
(a) The target and the formation centre trajectories



(b) The setpoints of *uav1*



(c) Overall view (all trajectories)



(d) The mutual distances of the neighbours

Figure 14: The experiment with 3 UAVs, *the mixing* algorithm, $w = 0.0$, $\beta = 1.5$, moving upwards on a slanted plane

4 Hardware and Software Description

4.1 Simulator

For simulations, the Gazebo multi-robot simulator[3], version 7.0.0, was used. Simulations were carried out on two desktop computers connected via a router, each with Ubuntu 16.04.4 LTS GNU/Linux operating system and the Kinetic Kame version of ROS[4]. One PC is equipped with 8-core Intel Core i7-7700 3.60GHz CPU and of 16 GB of RAM, the other with 4-core Intel Core i5-4570 3.20GHz CPU and 8GB RAM.

The advantage of using the ROS is particularly the high modularity, allowing usage of the same programe for both the simulations and experiments in the real world.

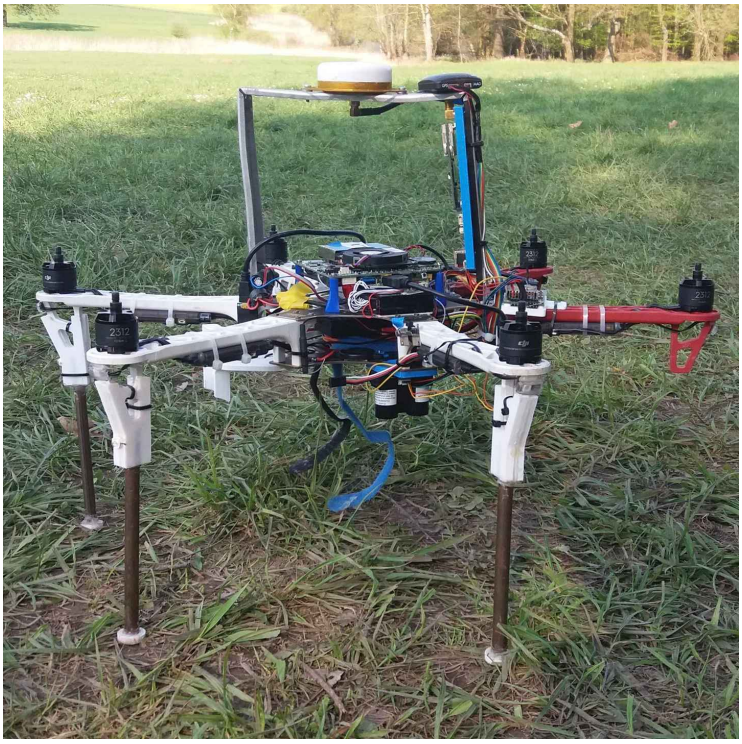


Figure 15: Overall view of one of MRS UAVs

[3] `http://gazebosim.org/`
[4] Robot Operating System, `http://www.ros.org/`

4.2 The UAVs

For real world experiments, the drones of the MRS group were used, see figure 15. They are based on the DJI Flame Wheel 550 construction[5] with 6 propellers. For a view from the above (29), see the appendix C where other photographs are available.

Modules on the drone include the Intel NUC onboard computer (5th generation Core-i7, 8 GB of RAM)[16], a camera (more in subsection 4.3), a ZigBee antenna, a GPS receiver, a WiFi client, and the Pixhawk embedded controller [2]. A 5GHz WiFi access point was stationary located in an altitude of approximately 3 metres, through which the UAVs were communicating to each other. All the communication was mediated by `multimaster`[6]. Further information about the platform can be found on the web page of the MRS group [7].

4.3 The Cameras

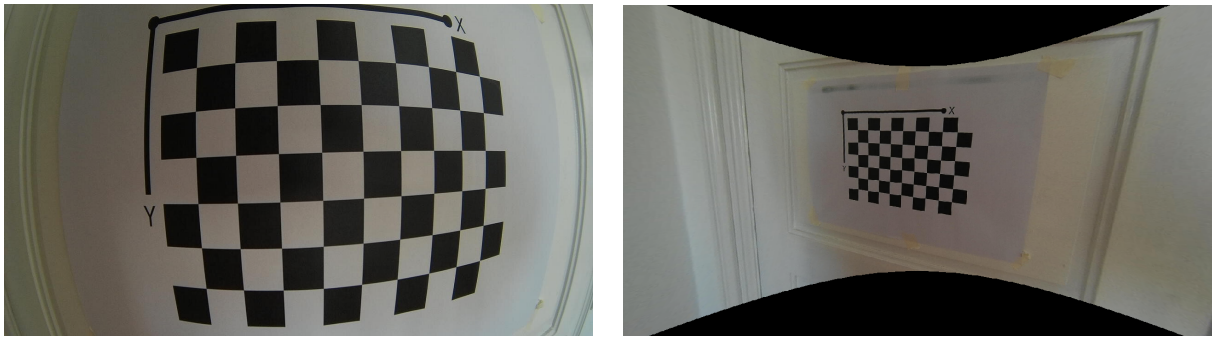
As a vision sensor, the Mobius 1 ActionCam monocular camera was used. It is capable of producing a 5–60 FPS video in a resolution of 1280×720 px. It was connected to the on-board computer with a USB cable. The camera mounted on one of the UAVs can be seen in figure 30.

Figure 16: The non-id Whycon pattern mounted on the UGV

[5] `https://www.dji.com/flame-wheel-arf/spec`
[6] `http://wiki.ros.org/multimaster`
[7] `http://mrs.felk.cvut.cz/mbzirc2019proposal`

4.4 Camera Calibration

In the experiments, new cameras were used. From the camera output, an appreciable distorition effect was notioned (see figures 17a and 17b). On that account, the camera calibration was carried out to update the configuration files containing information for the `camera_info` topic, and the `undistortPoints()`[8] function from the OpenCV library was added into the node `velecto1_copter` to undistort the received position of the target from the `whycon` node.



(a) An original photograph of the checkerboard (b) The undistorted photograph

Figure 17: The camera distortion

The camera calibration was carried out using the `cameracalibrator.py` node from the ROS package `camera_calibration`[9] according to the tutorial on ROS Wiki page[10] with a standard checkerboard.

Image resolution is 1280×720 px. The camera matrix:

$$\mathbb{C}_{ci} \doteq \begin{bmatrix} 652.245476 & 0 & 632.510830 \\ 0 & 736.408955 & 356.789219 \\ 0 & 0 & 1 \end{bmatrix} \tag{57}$$

The distortion coefficients:

$$\mathbb{D}_{ci} \doteq \begin{bmatrix} -0.291999 & 0.055179 & 0.002330 & -0.001738 & 0.000000 \end{bmatrix} \tag{58}$$

The projection matrix:

$$\mathbb{P}_{ci} \doteq \begin{bmatrix} 409.210114 & 0 & 628.670987 & 0 \\ 0 & 679.166016 & 357.901673 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{59}$$

[8] `https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html#undistortpoints`

[9] `http://wiki.ros.org/camera_calibration`

[10] `http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration`

4.5 Target

As the ground target the Cameleon UGV[11] was used. Its dimensions are in table 1.

Table 1: The Cameleon UGV dimensions

dimension	**value [cm]**
length	67
width	52
height	19

Some of the Whycon versions[12] use only a simple circular patter, some are able to detect multiple patterns having bars in additional inner circle (see figure 18), assigning an ID to each. As mentioned in section 2.6, only a non-id version was used.

Figure 18: A Whycon pattern with ID information

A non-id Whycon pattern was printed with the outer radius of the black circle being 40 cm. It was mounted on the UGV as seen in figures 16, 31, and 32. With the mounted pattern, the height of the UGV was approximately 50 cm in the front and 45 cm in the back.

[11] `https://www.ecagroup.com/en/solutions/cameleon-e-ugv-unmanned-ground-vehicle`
[12] `https://github.com/lrse/whycon`

5 Experimental Results

In the real-world experiments, the camera simulator and of the target detection using the Whycon system were tested. *The setpoint mixing* algorithm and *the exact* algorithm were used for this purpose. The formation behaviour was not tested in a real-world experiment due to technical difficulties.

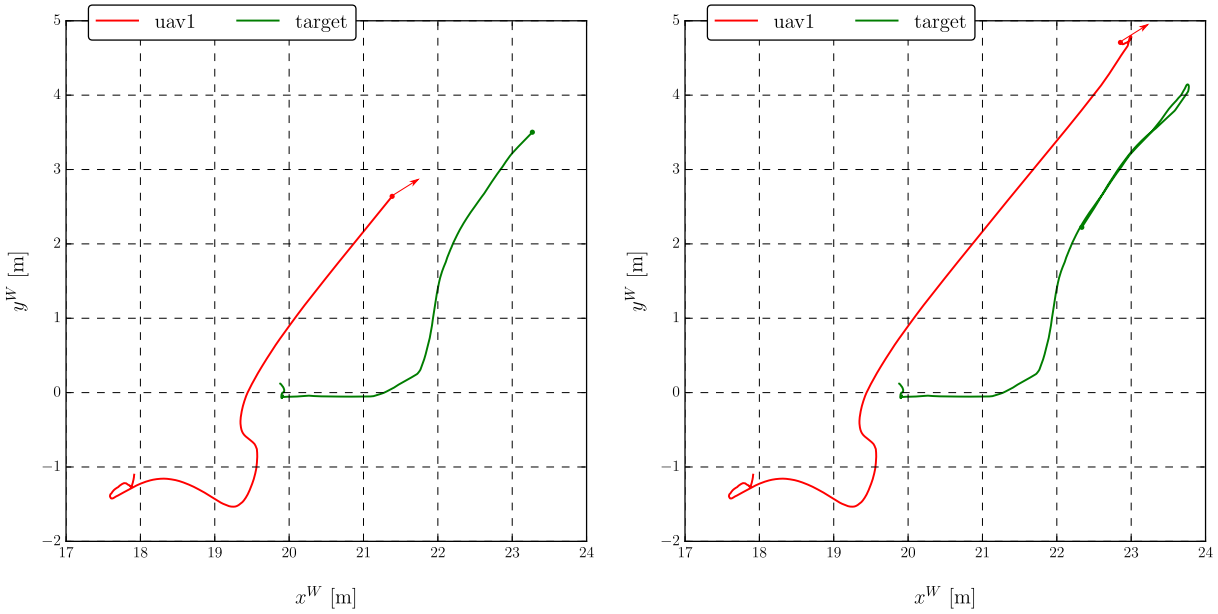
5.1 Target Following with One UAV

Figure 19: Following of another UAV

5.1.1 The Setpoint Mixing Algorithm with a UAV as the Target

The figure 19 shows the view of the observing UAV when following another UAV.

The trajectories observed in this experiment are shown in figure 20. The arrow denotes the heading of the observing UAV. The figure 20b shows the whole trajectories, the 20a shows a state in the course. In the 20a, the target is in the field of view of the observing UAV, while at the end, the target is lost (null setpoints can be seen in the figure 22 after $t = 10$ s). In the figure 21, coordinates of the error of the ideal position of the observing UAV from the target are shown in time.



(a) A state of the UAV and the target (b) A situation after the target turn

Figure 20: The trajectories in the first experiment

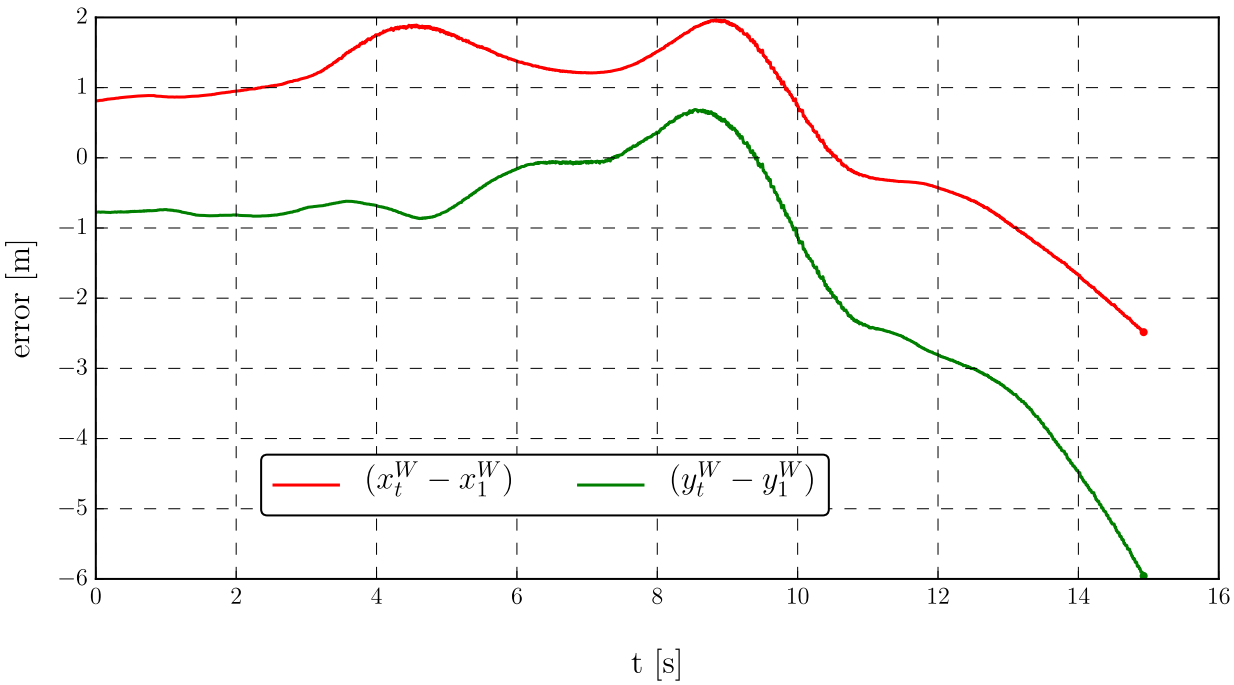


Figure 21: The error in x^W and y^W coordinates from the ideal position of the target

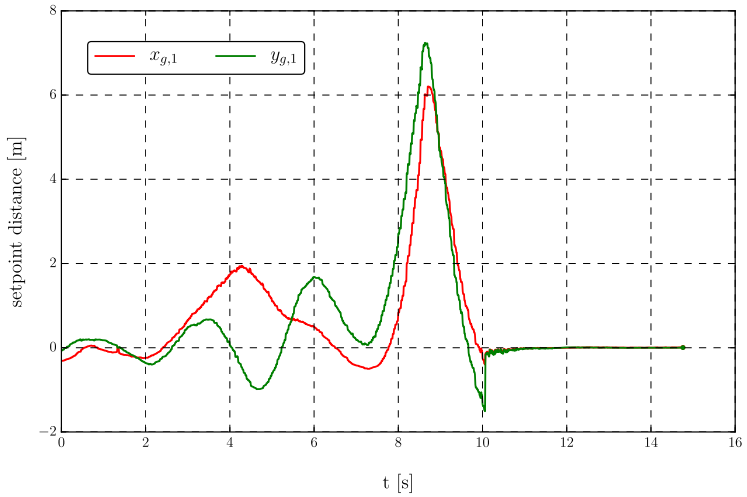


Figure 22: The setpoint coordinates in time

Figure 23: A photograph of the inital state with 3 UAVs

Figure 24: Following of the Cameleon UGV

5.1.2 The Setpoint Mixing Algorithm with the UGV as the Target

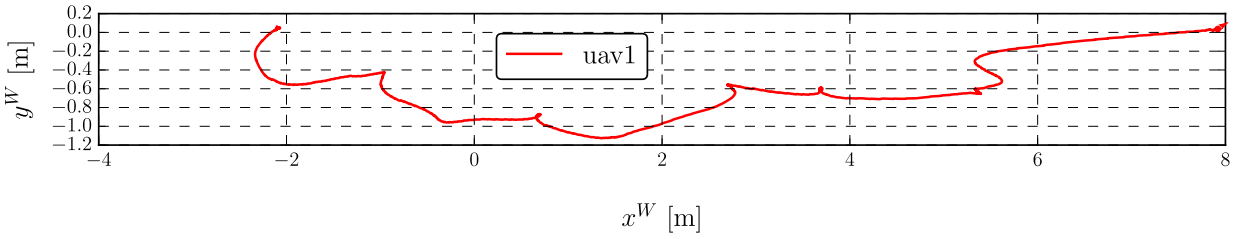


Figure 25: The trajectory of the observing UAV

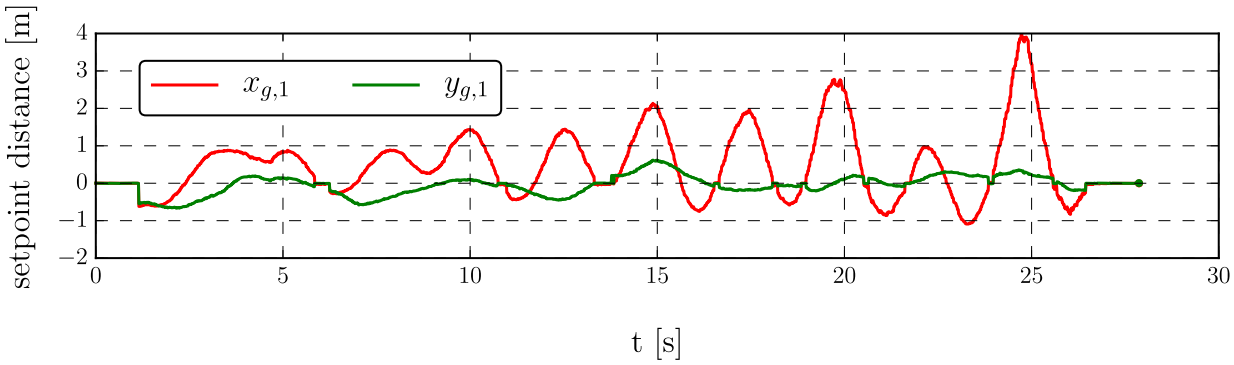


Figure 26: The setpoints coordinates in time

The course of this experiment is plotted in figures 25 and 26. The UGV in this experiment covered approximately 10 meters roughly on a line with a changing velocity. Oscillations of the UAV are visible. The view from the UAV is shown in the figure 24.

5.1.3 The Exact Algorithm with the UGV as the Target

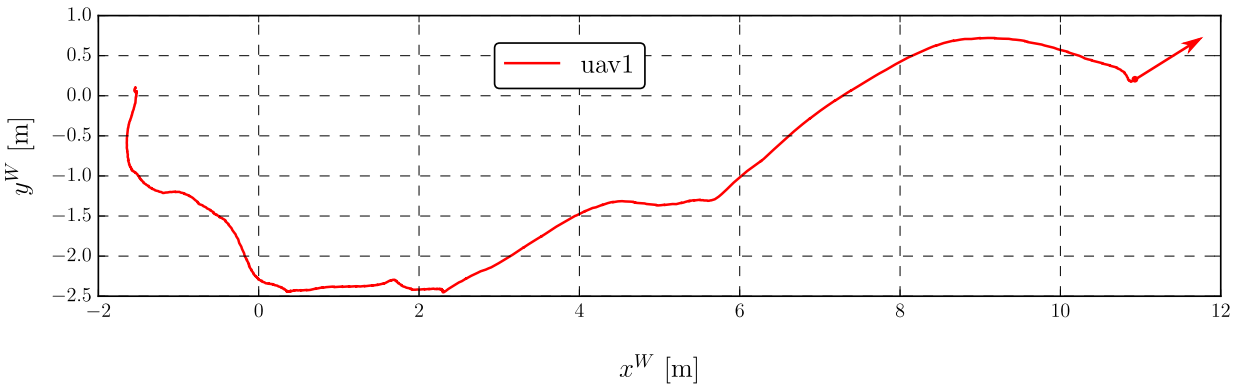


Figure 27: The trajectory of the observing UAV

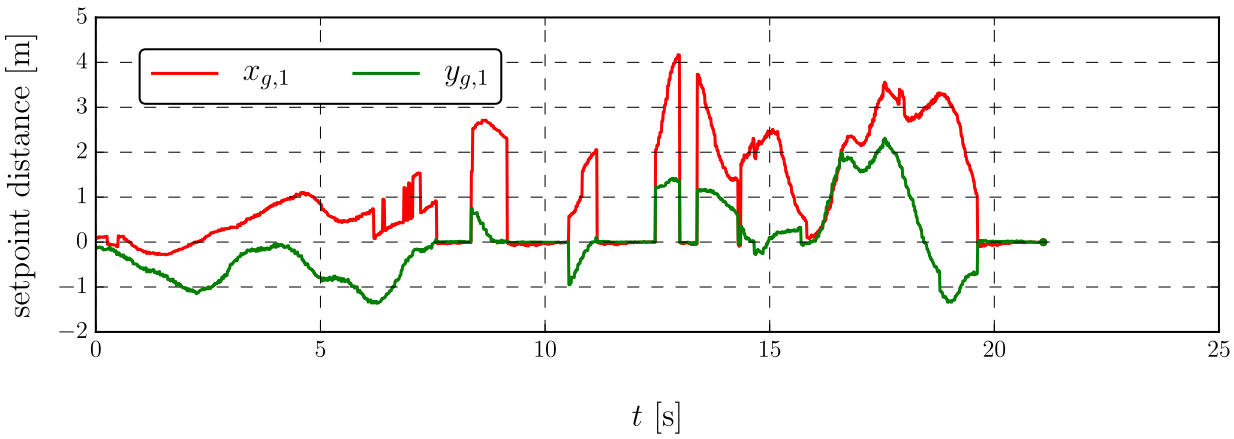


Figure 28: The setpoints coordinates in time

In this experiment, the UGV has covered a similar trajectory to the previous experiment. The trajectory of the UAV is plotted in the figure 27 and its setpoints are shown in 28. By watching the recorded videos of the onboard camera, one may say that this algorithm and its parameters leads to less smooth behaviour of the UAV in comparison with the previous one. It is visible in the setpoints plots — while both the algorithms cause oscillations, the setpoints plot of the former one is harmonic-shaped, the plot of the latter one contains an amount of sharp edges. On the other hand, while the trajectory of the UAV controlled by the former algorithm contains small loops, the trajectory plot of the latter one is smoother.

6 Conclusion

The algorithm for following a moving target described by F. Poiesi and A. Cavallaro in the article [1] was successfuly implemented for the UAV platform of the MRS group. Albeit there is a number of notable modifications, the principles — namely the intersection rule and the predicted velocity communication — were preserved.

To achieve a higher clarity and modularity, another layer was added to the original algorithm, creating *the exact algorithm*. That takes into account the effects of tilting, common in movement of the multirotor UAVs.

In order to allow all the UAVs in the formation to move smoothly, *the mixing algorithm* was proposed. It transforms the binary choice of behaviour from the original algorithm into a weighted average of the target-following and the formation-preserving actions.

The functionality of all the three algorithms to follow a moving target was proven in experiments in the simulated environment. In the simulations, the Whycon system was used successfully for both the target detection and for the detection of the neighbours.

The results of the experiments with the individual UAVs indicate that running any of the algorithms in the real world should be possible using the odometry data and even with the target detection using the Whycon system or a similar vision-based detector.

Due to technical difficulties, only limited experimentation has been performed in real-world environment. These tests have shown that for the algorithms to work with a pure vision-based localization of both the target and the neighbours, a different camera configuration would likely be necessary to modify the field of view of the UAV accordingly.

Additionally, in order to address the drawbacks stemming from the assumption of a completely flat terrain for the original algorithm, a method for a ground inclination estimation based on observed relative positions of the neighbouring UAVs with fixed vertical distances from the ground has been proposed.

Bibliography

[1] F. Poiesi and A. Cavallaro. Distributed vision-based flying cameras to film a moving target. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2453–2459, Sept 2015.

[2] Tomáš Báča, Petr Štěpán, and Martin Saska. Autonomous landing on a moving car with unmanned aerial vehicle. In *ECMR*, 2017.

[3] T. Krajník, M. Nitsche, J. Faigl, T. Duckett, M. Mejail, and L. Přeučil. External localization system for mobile robotics. In *16th International Conference on Advanced Robotics (ICAR)*, Nov 2013.

[4] Tomáš Krajník, Matías Nitsche, Jan Faigl, Petr Vaněk, Martin Saska, Libor Přeučil, Tom Duckett, and Marta Mejail. A practical multirobot localization system. *Journal of Intelligent & Robotic Systems*, 2014.

[5] Matias Nitsche, Tomáš Krajník, Petr Čížek, Marta Mejail, and Tom Duckett. Whycon: An efficent, marker-based localization system. In *IROS Workshop on Open Source Aerial Robotics*, 2015.

[6] Jan Faigl, Tomas Krajnik, Jan Chudoba, Libor Preucil, and Martin Saska. Low-cost embedded system for relative localization in robotic swarms. In *International Conference on Robotics and Automation (ICRA)*, pages 993–998. IEEE, 2013.

[7] R. Fonseca and W. Creixell. Tracking and following a moving object with a quadcopter. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, Aug 2017.

[8] F. Lin, X. Dong, B. M. Chen, K. Y. Lum, and T. H. Lee. A robust real-time embedded vision system on an unmanned rotorcraft for ground target following. *IEEE Transactions on Industrial Electronics*, 59(2):1038–1049, Feb 2012.

[9] Z. Li and J. Ding. Ground moving target tracking control system design for uav surveillance. In *2007 IEEE International Conference on Automation and Logistics*, pages 1458–1463, Aug 2007.

[10] TS Jin, JW Park, and JM Lee. Trajectory generation for capturing a moving object in predictable environments. *Jsme International Journal Series C-Mechanical Systems Machine Elements and Manufacturing*, 47(2):722–730, 2004.

[11] A. Razinkova and H. C. Cho. Tracking a moving ground object using quadcopter uav in a presence of noise. In *2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1546–1551, July 2015.

[12] G. López-Nicolás, M. Aranda, and Y. Mezouar. Formation of differential-drive vehicles with field-of-view constraints for enclosing a moving target. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 261–266, May 2017.

[13] H. Cheng, L. Lin, Z. Zheng, Y. Guan, and Z. Liu. An autonomous vision-based target tracking system for rotorcraft unmanned aerial vehicles. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1732–1738, Sept 2017.

[14] Radim Šára. Homography, perspective camera. Lecture slides for *3D Computer Vision* course at CTU FEE, `http://cmp.felk.cvut.cz/cmp/courses/TDV/2017W/lectures/tdv-2017-02-annotated.pdf`, Feb 2017.

[15] OpenCV development team. *OpenCV documentation*, 2.4.13.6 edition, 2014. See `https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html`.

[16] Tomas Baca, Daniel Hert, Giuseppe Loianno, Martin Saska, and Vijay Kumar. Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles. `http://mrs.felk.cvut.cz/data/papers/ral2018mpctracker.pdf`, 2018.

Appendix A CD Content

In table 2, there are listed names of the most important directories on CD.

Table 2: CD Content

Directory name	**Description**
workspace_src/ros_nodes	ROS nodes sources
workspace_src/scripts	scripts for running simulations
workspace_src/sources	additional files
workspace_src/sources/Calibration_ROS	the Mobius camera configuration file
workspace_src/sources/Holder	OpenSCAD model of camera holder
workspace_src/sources/Screen_exponential	Matlab scripts which generated fig. 3
thesis	the thesis in pdf format
thesis_sources	LaTeX source codes
velecto1-uavproject	scripts used for experiments
whycon-mix	an ID version of whycon
models	the box with the pattern and real terraing model for Gazebo

Appendix B List of abbreviations

In table 3 are listed abbreviations used in this thesis.

Table 3: Lists of abbreviations

Abbreviation	Meaning
FOV	field of view
FPS	frames per second
GNSS	Global navigation satelite system
MPC	model predictive control
MRS	Multi-robot Systems (group)
PI	proportional-integral (controller)
PID	proportional-integral-derivative (controller)
ROS	Robot Operating System
RTK	real-time kinematic
UAV	unmanned aerial vehicle
UGV	unmanned ground vehicle

Appendix C Photographs of the used platforms

Figure 29: The MRS drone from above

Figure 30: The Mobius camera and its holder from profile

Figure 31: Profile of the cameleon UGV

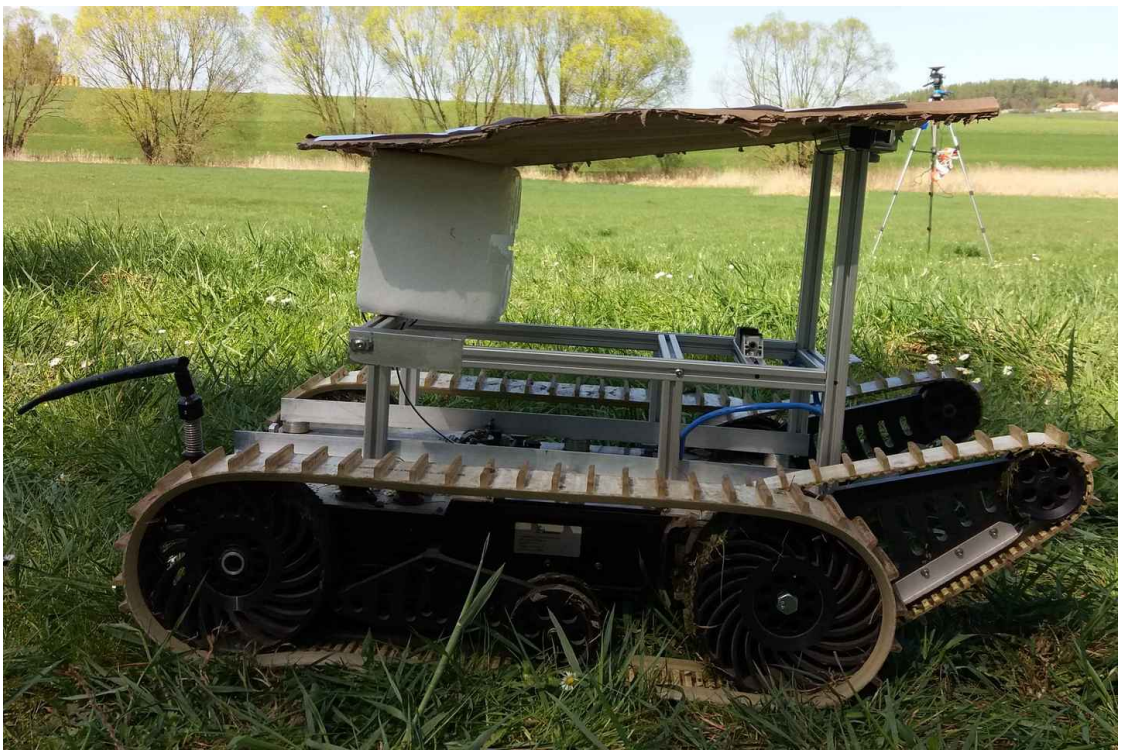


Figure 32: A side view of the cameleon UGV

Appendix D Configuration Files Used in the Experiments

The configuration file parameters used in the first real-world experiment (section 5.1.1) is shown in the figure 33. A newer version, used for the experiment shown in the figure 37, is in the figure 35. The explanations of the parameter names is summarized in the table 4.

```
formationSize: 1
formationAltitude: 7.0
tau_a: 20
rate: 30
c11: 80000
c12: 0
c21: 0
c22: 24000
samplingTime: 333.33
defaultDistance: 4
distanceThreshold: 0.8
alpha: 1.0
beta: 1.0
formationShapeWeight: 0.5
neighbourPositionType: 'odometry'
targetPositionType: 'fake'
whyconSet: 'all'
controlInput: 'relativeDifference'
controlOutput: 'relativeMove'
addNoise: no
neighboursPxOffset: 100
neighboursMOffset: 2
NNpxRadius: 80
NNmRadius: 1
delayThreshold: 0.5
odometryDelay: 0
cx: 0.211
cy: 0.0
cz: -0.05
croll: 0.0
cpitch: 0.7
cyaw: 0.0
noDoZone: 0.1
trajectoryLength: 1
useTargetKalman: no
useNeighboursKalman: no
useLASER: no
```

Figure 33: The version of the configuration file used in the experiment 5.1.1

Table 4: Explanation of the parameters of the node

Parameter name	**Notation / Explanation**
`formationSize`	N [-] (the number of UAVs in the formation)
`formationAltitude`	h_i [m] (the distance which should the UAVs keep from the ground)
`cx`, `cy`, `cz`	the coordinates [m] defining the translation $\mathbb{T}^{UAV}_{camera}$
`croll`, `cpitch`, `cyaw`	the Euler angles [rad] defining the rotation $\mathbb{R}^{UAV}_{camera}$
`whyconSet`	what data to take from the Whycon system: both for the target and for the neighbours `all`, or for the target only (`target_only`)
`controlInput`	switches between methods of $\boldsymbol{a}$ calculation (`cameraCoords` for the *original* algorithm, and `relativeDifference` for both the *exact* and the *setpoint mixing* algorithms)
`controlOutput`	the algorithm choice: `velocity` for the original and for the exact algorithm, `relativeMove` for the *mixing*
`intersectionRule`	`original` or `naive` (for small formations)
`useTargetKalman`	a trigger for smoothing and prediction of the position of the target using a Kalman filter (it was not tested properly, therefore is not described)
`useNeighboursKalman`	analogical
`useCorrectedTarget`	a trigger for refining of the target position
`undistortTarget`	a trigger for undistortion of the coordinates
`rate`	the processing speed [Hz]
`tau_a`	τ_a
`c11`, `c12`, `c21`, `c22`	the Σ_m matrix elements
`defaultDistance`	a [m]
`distanceThreshold`	ϵ [m]
`alpha`	α
`noDoZone`	the deadband [m]
`formationShapeWeight`	w [-]
`yawDifferenceConstant`	γ [-]
`distanceTimeout`	the timeout for the formation shape verification [s]
`maxVelocity`	v_{max} [m/s]
`beta`	β [-]
`expFiltConst`	q [-]
`targetPositionType`	how the target coordinates are obtained
`neighbourPositionType`	how the relative positions of the neighbours are obtained
`neighboursPxOffset`	a value used in the grouping of the data from the Whycon nodes; the `neighboursMOffset`, `NNpxRadius`, and `NNmRadius` are similar
`delayThreshold`	a threshold for ignoring the outdated data

```yaml
# FORMATION DESCRIPTION
formationSize: 3
formationAltitude: 7.0
supposedInitialHeight: 3.0
targetName: 'uav5'

# DRONE SETTINGS DESCRIPTION
cx: 0.22
cy: 0.0
cz: -0.07
croll: 0.0
cpitch: 0.7
cyaw: 0.0

# CONTROL TYPE
whyconSet: 'target_only'
controlInput: 'cameraCoords'
controlOutput: 'velocity'
intersectionRule: 'naive'
useTargetKalman: no
useNeighboursKalman: no
useCorrectedTarget: no
undistortTarget: no

# ORIGINAL ALGORITHM CONSTANTS
rate: 25
tau_a: 5
c11: 80000
c12: 0
c21: 0
c22: 24000
defaultDistance: 10
distanceThreshold: 2.0
alpha: 1.5

# ADDITIONAL ALGORITHM PARAMETERS
noDoZone: 0.0
formationShapeWeight: 0.5
yawDifferenceConstant: 0.13
partialMoveRatio: 0.5
distanceTimeout: 0.3
maxVelocity: 100.0
beta: 1.0
expFiltConst: 0.2

# SIMULATION PARAMETERS
targetPositionType: 'fake'
neighbourPositionType: 'odometry'
addNoise: no

# TRACKER PARAMETERS
neighboursPxOffset: 100
neighboursMOffset: 2
NNpxRadius: 80
NNmRadius: 1
delayThreshold: 0.8
odometryDelay: 0
```

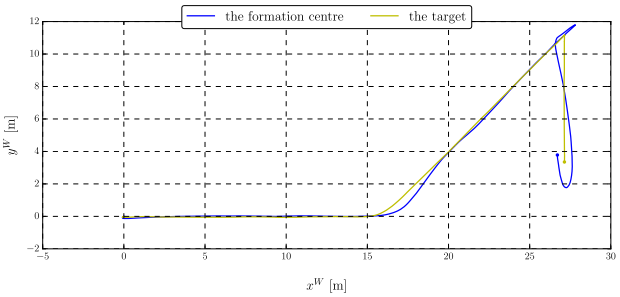
Figure 34: The version of the configuration file used in the experiment 11

```
# FORMATION DESCRIPTION
formationSize: 3
formationAltitude: 7.0
supposedInitialHeight: 3.0
targetName: 'uav5'

# DRONE SETTINGS DESCRIPTION
cx: 0.22
cy: 0.0
cz: -0.07
croll: 0.0
cpitch: 0.7
cyaw: 0.0

# CONTROL TYPE
whyconSet: 'target_only'
controlInput: 'cameraCoords'
controlOutput: 'velocity'
intersectionRule: 'naive'
useTargetKalman: no
useNeighboursKalman: no
useCorrectedTarget: no
undistortTarget: no

# ORIGINAL POIESI ALGORITHM CONSTANTS
rate: 25
tau_a: 5
c11: 80000
c12: 0
c21: 0
c22: 24000
defaultDistance: 10
distanceThreshold: 2.0
alpha: 1.5

# ADDITIONAL ALGORITHM PARAMETERS
noDoZone: 0.0
formationShapeWeight: 0.5
yawDifferenceConstant: 0.13
partialMoveRatio: 0.5
distanceTimeout: 0.3
maxVelocity: 100.0
beta: 1.0
expFiltConst: 0.2

# SIMULATION PARAMETERS
targetPositionType: 'fake'
neighbourPositionType: 'odometry'
addNoise: no

# TRACKER PARAMETERS
neighboursPxOffset: 100
neighboursMOffset: 2
NNpxRadius: 80
NNmRadius: 1
delayThreshold: 0.8
odometryDelay: 0
```

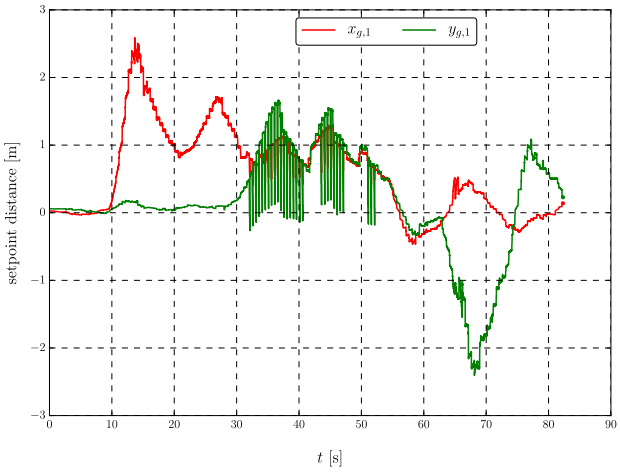
Figure 35: The version of the configuration file used in the experiment 37

Appendix E Plots from the Simulations

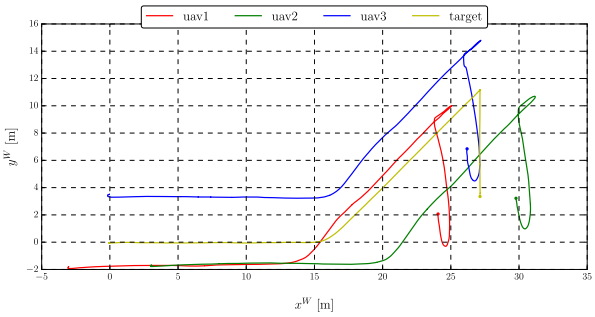
In the last pages, there are the remaining figures from the simulated experiments.



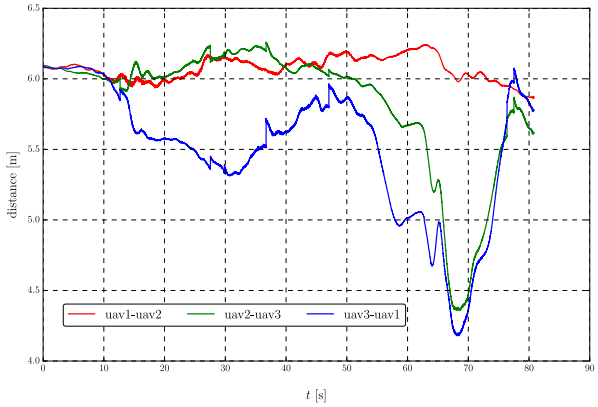
(a) The target and the formation centre trajectories



(b) The setpoints of *uav1*

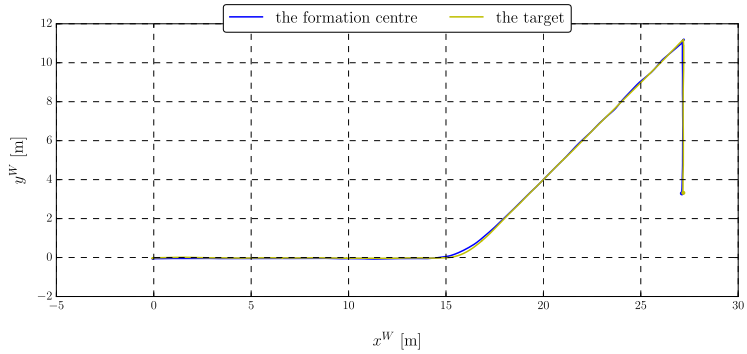


(c) Overall view (all trajectories)

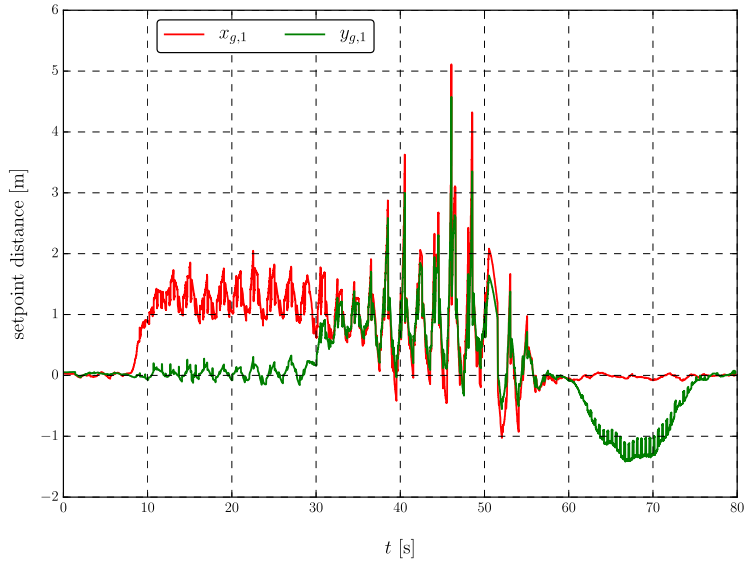


(d) The mutual distances of the neighbours

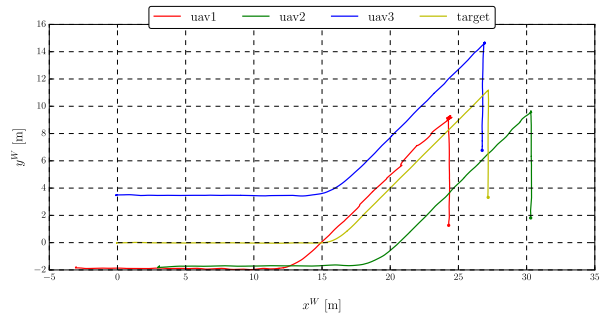
Figure 36: The experiment with 3 UAVs, *the exact* algorithm, $\alpha = 1.25$, $\beta = 1.5$



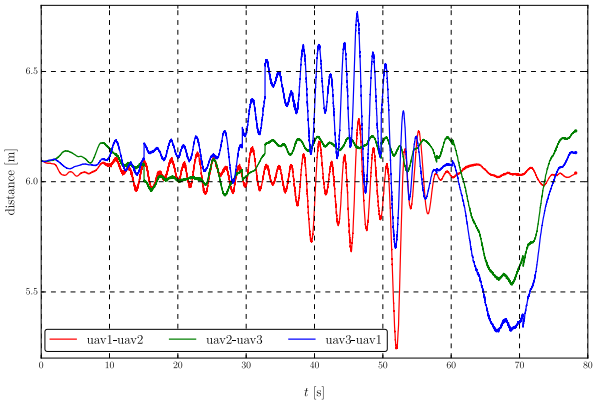
(a) The target and the formation centre trajectories



(b) The setpoints of *uav1*

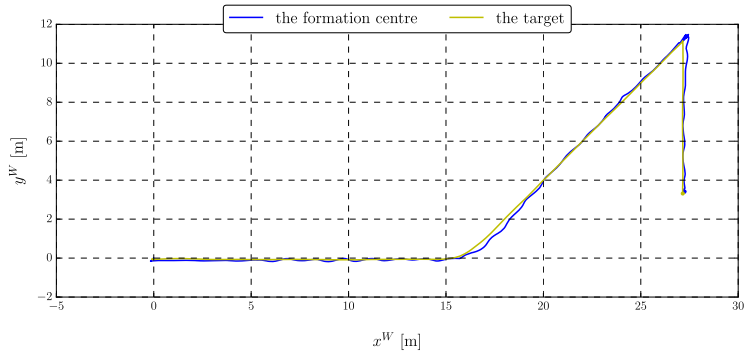


(c) Overall view (all trajectories)

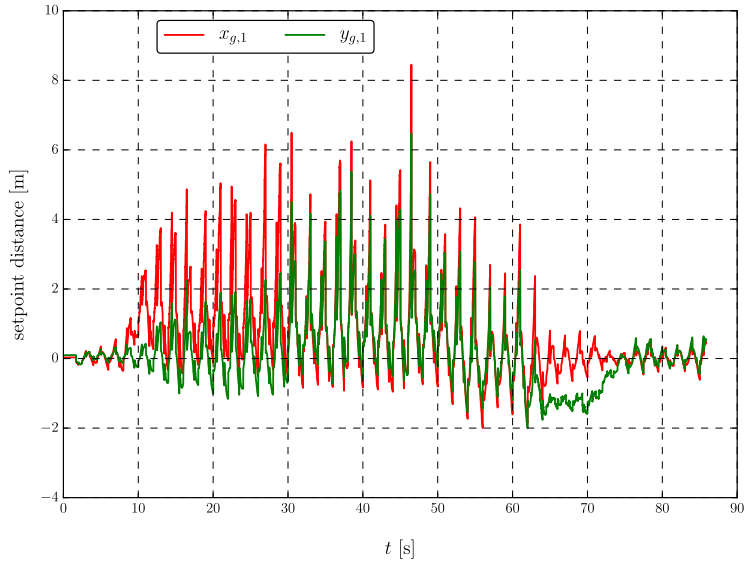


(d) The mutual distances of the neighbours

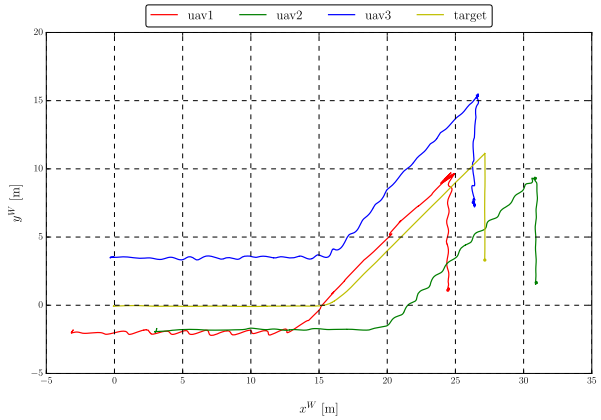
Figure 37: The experiment with 3 UAVs, *the mixing* algorithm, $w = 0.0$, $\beta = 1.5$



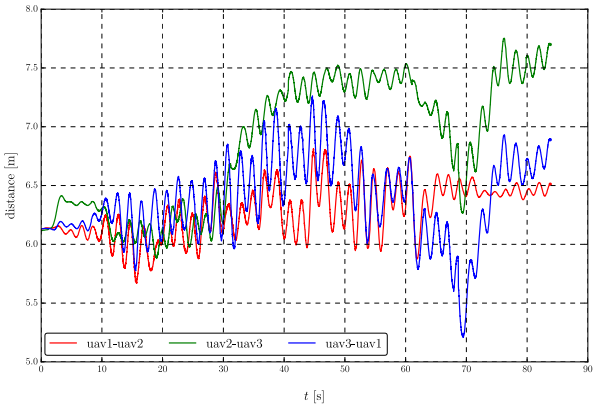
(a) The target and the formation centre trajectories



(b) The setpoints of *uav1*

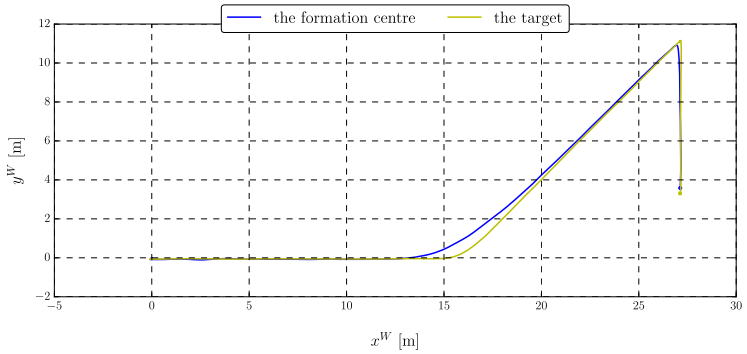


(c) Overall view (all trajectories)

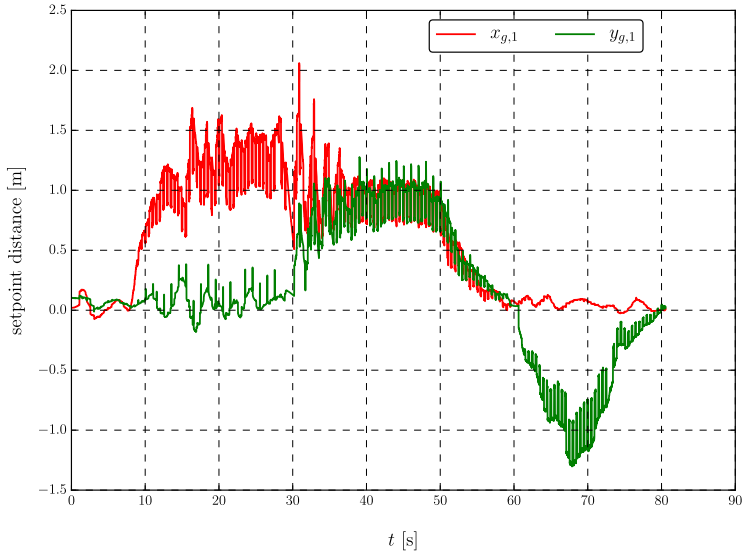


(d) The mutual distances of the neighbours

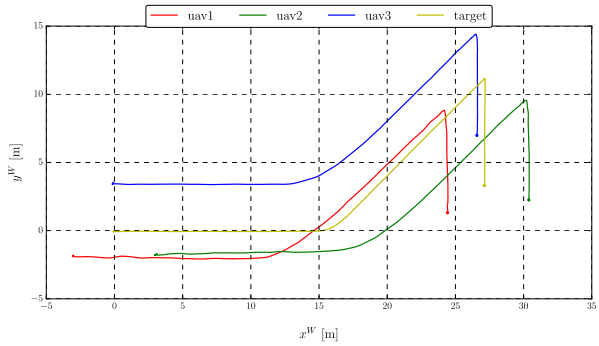
Figure 38: The experiment with 3 UAVs, *the mixing* algorithm, $w = 0.5$, $\beta = 1.5$



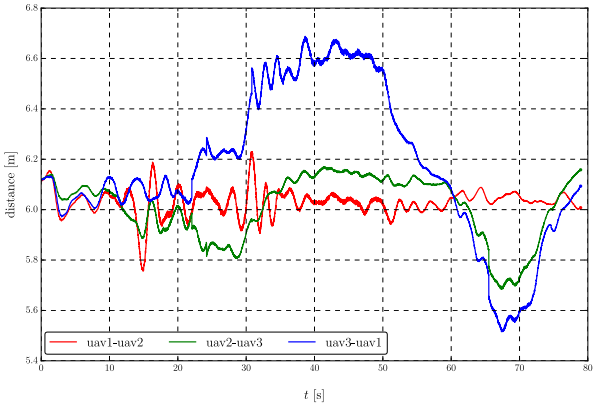
(a) The target and the formation centre trajectories



(b) The setpoints of *uav1*

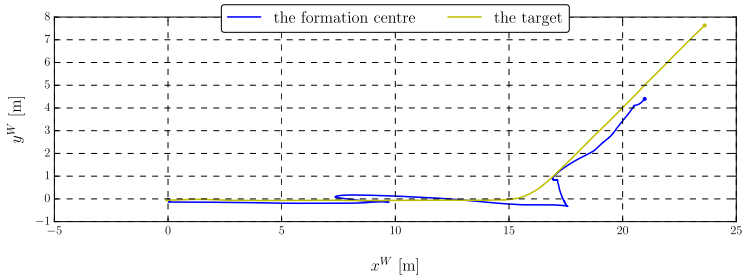


(c) Overall view (all trajectories)

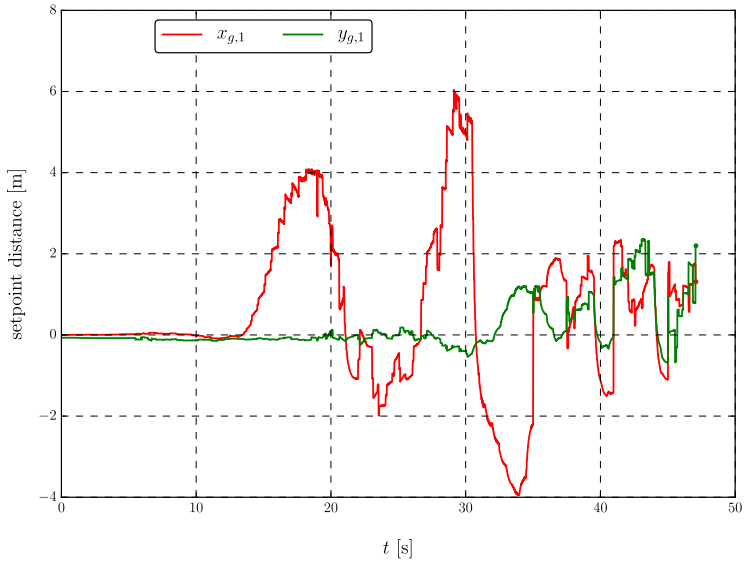


(d) The mutual distances of the neighbours

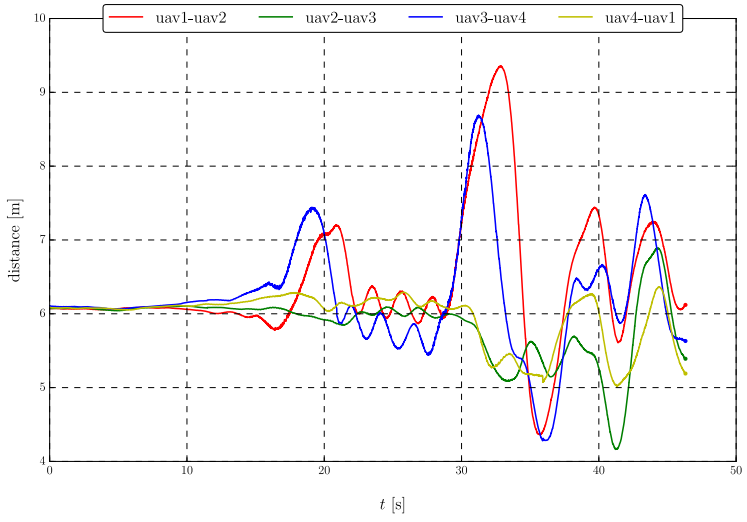
Figure 39: The experiment with 3 UAVs, *the mixing* algorithm, $w = -0.5$, $\beta = 1.5$



(a) The target and the formation centre trajectories

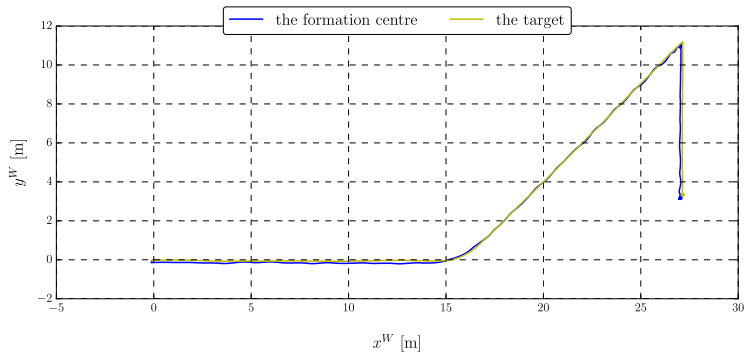


(b) The setpoints of *uav1*

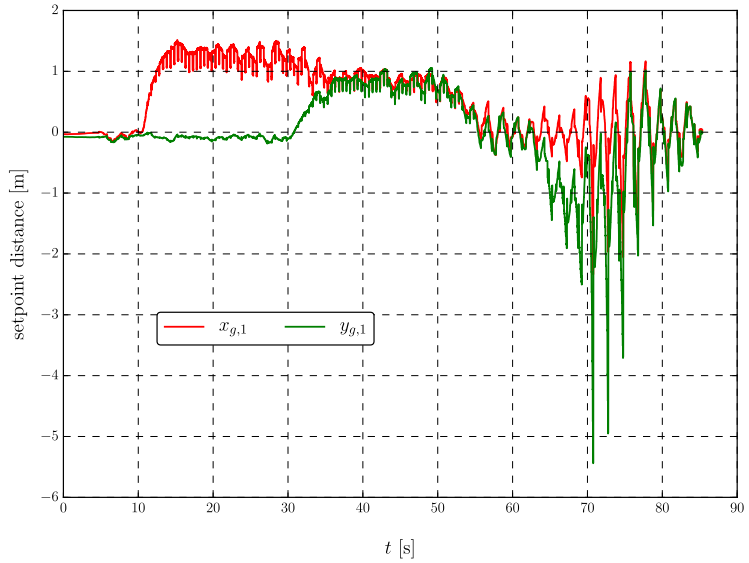


(c) The mutual distances of the neighbours

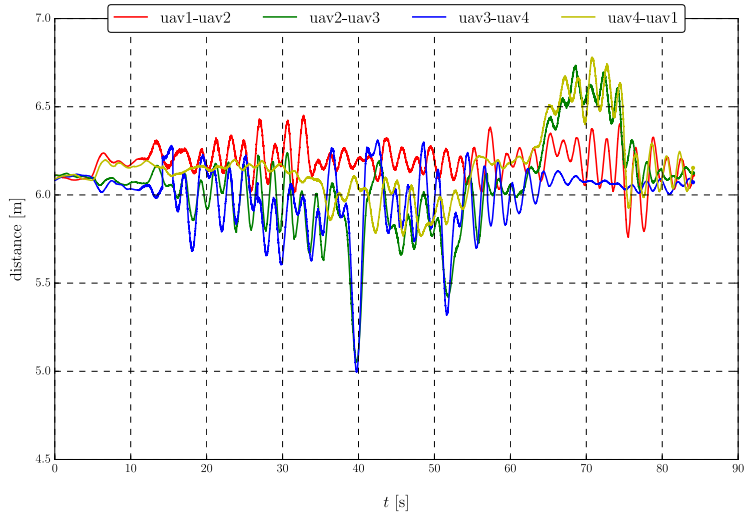
Figure 40: The experiment with 4 UAVs, *the exact* algorithm, $\alpha = 1.0$, $\beta = 2.0$



(a) The target and the formation centre trajectories

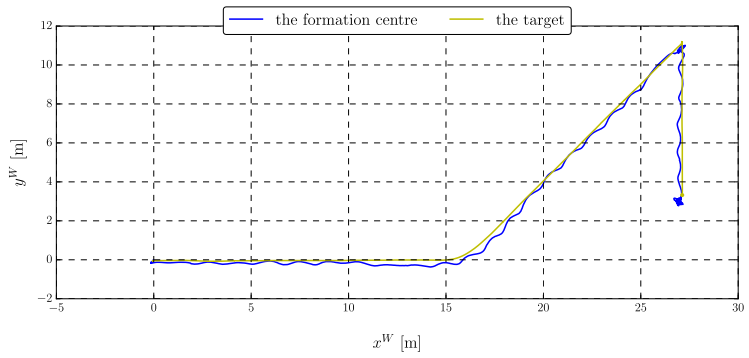


(b) The setpoints of *uav1*

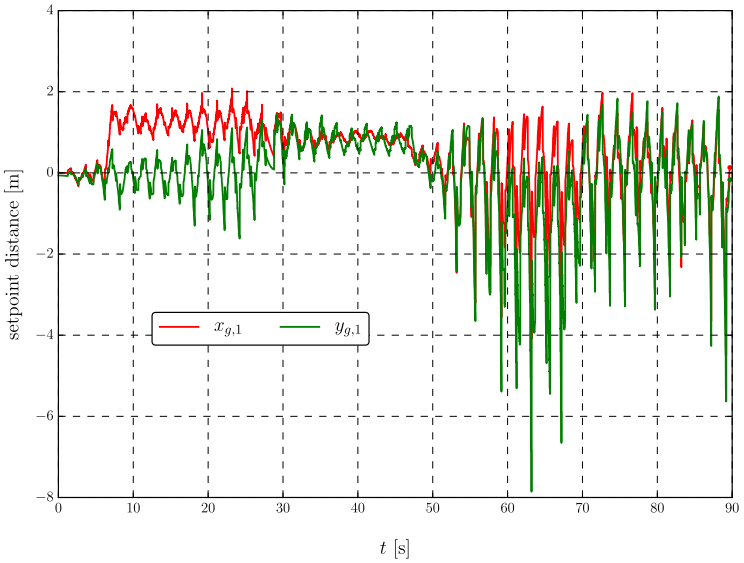


(c) The mutual distances of the neighbours

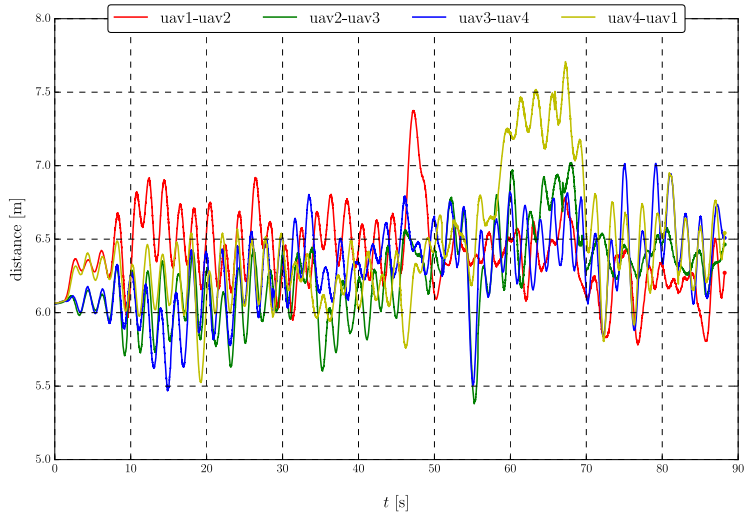
Figure 41: The experiment with 4 UAVs, *the mixing* algorithm, $w = 0.0$, $\beta = 1.5$



(a) The target and the formation centre trajectories



(b) The setpoints of *uav1*



(c) The mutual distances of the neighbours

Figure 42: The experiment with 4 UAVs, *the mixing* algorithm, $w = 0.5$, $\beta = 1.5$