Chapter 15
A Multi Agent System for Precision Agriculture

Amélie Chevalier, Cosmin Copot, Robin De Keyser, Andres Hernandez and Clara Ionescu

Abstract This chapter investigates the use of a multi agent system for precision agriculture. Precision agriculture refers to the management of farm operations based on observation, measurement and fast response to inter- and intra-field variability in crops. Important characteristics in this application are path following, disturbance rejection and obstacle avoidance of which path following is addressed in this chapter. This study combines the degree of freedom of aerial vehicles with the location precision of ground vehicles in order to achieve a common goal. The multi agent system in this study consists of a quadrotor as the aerial agent and tracked robots with a rotary camera as the agents on the ground to achieve the common task of following a predefined path while maintaining the formation. This research uses a combination of low-level PID cascade control for the ground vehicles with a high level predictive control for the aerial agent to ensure optimal control of the positions of the ground robots and the quadrotors. A series of proof-of-concept experiments for this novel combined control strategy are performed. Simulation and experimental results suggest that the proposed control system is able to maintain the formation of the ground vehicles and provide a good tracking of the ground vehicles by the quadrotor. Because of the setup of the described system, it has also potential applications in traffic control and analysis, search and rescue operations, etc.

A. Chevalier (✉) · C. Copot · R. De Keyser · A. Hernandez · C. Ionescu
Department of Electrical energy, Systems and Automation, Research Group on Dynamical Systems and Control, Ghent University, Technologiepark 914, 2nd floor, 9052 Ghent, Belgium
e-mail: Amelie.Chevalier@ugent.be

C. Copot
e-mail: Cosmin.Copot@ugent.be

R. De Keyser
e-mail: Robain.DeKeyser@ugent.be

A. Hernandez
e-mail: Andres.Hernandez@ugent.be

C. Ionescu
e-mail: Claramihaela.Ionescu@ugent.be

L. Buşoniu and L. Tamás (eds.), *Handling Uncertainty and Networked Structure in Robot Control*, Studies in Systems, Decision and Control 42,
DOI 10.1007/978-3-319-26327-4_15

15.1 Introduction

With robots becoming irreplaceable in daily life, there was a need for them to become more universal and functional. Several attempts were made to have them perform a wide range of tasks with sufficient accuracy and speed. The first efforts resulted in highly technological, yet very expensive and unwieldy robots. To counteract these drawbacks, a less expensive and more robust option was found in Multi-Agent Systems (MAS), i.e. a system that entails the working together of several simple, accurate and fast operating robots to achieve a common goal (Cao et al. 2013). Compared to the use of a single robot, MAS have the advantage of being more robust, flexible and adaptable. These benefits and the high potential for an enhanced performance that goes with it, has sparked an interest in the coordinated control of autonomously operating mobile robots.

Each member of a MAS is an unmanned autonomous system, specialized in performing one or more specific tasks and enhancing the flexibility of the overall system. The complete MAS has a specific goal and can be adapted to certain specifications. The ability to interact with a changing environment is of prime importance. The agents in a MAS should be able to anticipate and react to real-life events and thus requiring formation control, formation repair and obstacle avoidance, for which higher level decision making becomes vital.

Nowadays, MAS are omnipresent in all sorts of applications. Examples of researched MAS are railway surveillance and maintenance as could be derived from Páll et al. (2014), search and rescue missions (Beard et al. 2013) and MAS for the analysis of traffic dynamics such as highway bottlenecks (Chevalier et al. 2013a) and stop-and-go waves (Chevalier et al. 2013b; Caruntu et al. 2014).

As MAS are in continuous development, they have to deal with many new problems and challenges arising in the process of evolution. As a result, the study of formation control emerged from the development of MAS and has been the subject of extensive research (Guanghua et al. 2013).

In this research a decentralized formation control approach is used with the advantage of flexibility from an operating point-of-view. A leader-follower strategy will be used in which the controllers are designed in the frequency domain by means of FRTool (De Keyser and Ionescu 2006) as can be seen in the work of Chevalier et al. (2013a). The used MAS consists of a group of ground vehicles in combination with a quadrotor as aerial agent. The novelty of this approach lies in the added flexibility of the ground vehicles by adding a rotary camera and using the quadrotor for an aerial overview of the formation of the ground vehicles. The quadrotor has then the possibility to act as a flying sensor for the group of ground vehicles by noticing obstacles beforehand and providing this information to the ground vehicles.

The application area for this MAS is situated in precision agriculture. Precision agriculture (PA) is defined as the management of farm operations based on observation, measurement and fast response to inter- and intra-field variability in crops. The presented MAS can be used in PA as a highly flexible sensor system for measurement and observation. Formation control gives the ground vehicles the possibility to move

across the field without damaging the crops. The formation of the ground agents is adjusted to the width of the crop lines. The quadrotor acts as a flying sensor guiding them to the areas of interest in the field e.g. places of dehydration.

In what follows, the general architecture of the MAS is firstly described in section two. The third section addresses the modeling of the different agents after which the different control methodologies are discussed. The fourth section explains the experimental results while the fifth section contains the conclusions resulting from this research.

15.2 General Architecture

In this section a general overview of the architecture of the MAS and a background of each agent in the MAS is given in order to comprehend each unit of the MAS. For both the ground agents as well as the aerial agent, a description of the hardware and software is presented. The goal of this research is to develop a high-performance controller for the MAS.

The MAS in this study consists of a group of *unmanned ground vehicles* (UGVs) and a quadrotor which is the *unmanned aerial vehicle* (UAV) in the MAS. The goal of the system is to have the UGVs drive in a fixed formation defined by the operator. Therefore, a formation control strategy is discussed in subsequent sections. The quadrotor must be able to follow the group of UGVs. For this purpose, a path following control strategy will be developed for the quadrotor.

The group of *unmanned ground vehicles* (UGVs) at the center of this study is composed of Surveyor SRV-1 robots (Fig. 15.1), which have a quad-motor tracked robotic base with two tank-style treads. The processor dictates the motor control and the motor output pins via the on-board PWM interface. Each pair of DC motors can be controlled separately. In addition, optical wheel encoders are mounted on the side of the robot to measure the speed of the wheels. The on-board camera has been equipped with a stepper motor so that it can rotate. The rotary camera allows each robot to independently determine its position and orientation with respect to its leader. Consequently, the robots do not need communication between them and

Fig. 15.1 Surveyor SRV-1 robot with two tank-style treads. (Holmes 2014)

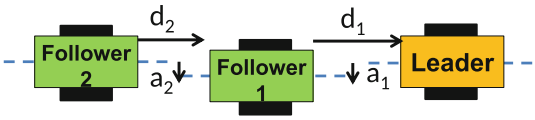


Fig. 15.2 *In-line* leader-follower formation with d_i the distance and a_i the lateral offset

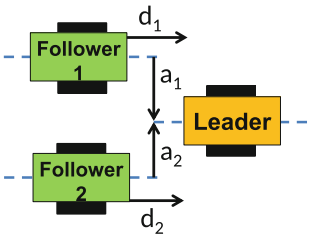


Fig. 15.3 *Triangular* leader-follower formation with d_i the distance and a_i the lateral offset

Fig. 15.4 The AR.Drone quadrotor. (Parrot 2014)

communication delays are avoided. Thus, robustness and speed of control can be increased. Moreover, it allows for an easy extension to a larger number of robots. We use a leader-follower formation where one robot is considered as the leader, while others act as followers. Two types of predefined formations are used: an *in-line* formation and a *triangular* formation. Both formations are used to keep the robots off the crop lines to avoid damaging the crops. The *in-line* leader-follower approach is presented in Fig. 15.2. In this configuration every robot is a follower of the preceding one and a leader for the next one. The lateral offsets a_1 and a_2 are set to 0 to have the *in-line* configuration. The *triangular* formation consists of one leader robot followed by two follower robots that have a predefined lateral offset from the center of the leader robot depending on the width of the crop lines (Fig. 15.3). Markers are used to make sure that each follower recognizes its corresponding leader. A leader has three markers: a red one at the back and a green one at each side. Based on color segmentation algorithms, the followers know whether they are on the left or the right of the leader.

The aerial agent in this study is the AR.Drone 2.0 which is a commercial and low-cost micro *unmanned aerial vehicle* (UAV) (Fig. 15.4). The quadrotor comes with internal in-flight controllers and emergency features making it possible to fly (Bristeau et al. 2011). The only downside would be that access to the internal controller of the quadrotor is restricted. The internal software is black-box and the parameters that refer to control, motors and other calibrations are undocumented. AR.Drone electronics execute an operating system to read the sensors, to manipulate the speed of the motors, and to control the quadrotor in four degrees of freedom. We refer to

Fig. 15.5 Quadrotor layers: The low layer represents the electronic assistance and the embedded operating system on the AR.Drone, the high layer represents the pilot

this black-box on-board system as the low layer. Our own path-following controller located in the higher layer, sends references to the low-layer internal controllers through Wi-Fi communication. Figure 15.5 describes the two layers structure which characterizes the system. Movement of the quadrotor is achieved by giving reference values as input to the internal, black-box controllers using a computer. The input and output relations will be discussed in the following subsection.

15.3 Methodology

To control the MAS, two methodologies are combined to create a hierarchical structure. This structure consists of a low-level PID cascade control for the formation of the UGVs and a high-level Extended Prediction Self-Adaptive Control (EPSAC) strategy for the tracking ability of the UAV in order to obtain an increased control performance. As the designed strategies for the controller are model-based, identification of the systems is also provided.

15.3.1 Model Identification

As the design techniques for the controllers of the MAS are model-based techniques, a model for each agent is needed. Identification techniques are used to obtain the transfer functions of each agent.

Ground Agents

To build a controller that satisfies the leader-follower formation control target, i.e. making the followers form and move according to a predefined geometrical pattern with respect to their leader, we firstly derive a theoretical model for the system under scrutiny. To do this, the Lagrange method will be used (Brizard 2008). Apart from gaining insight into the system, the derivation exercise ensures that the appropriate control approach is developed. Secondly, based on the footage from the camera, we will derive equations necessary to calculate the controlled variables. These equations will provide us with feedback in our control strategy.

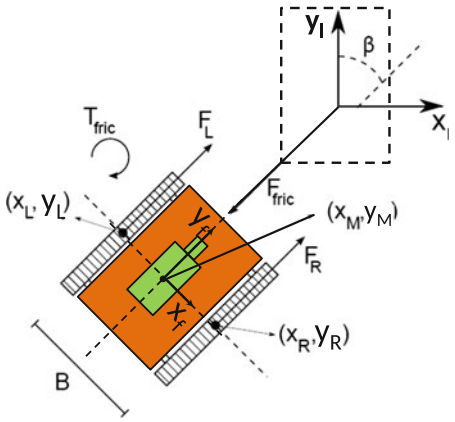


Fig. 15.6 Model of the system with a fixed coordinate frame is attached to both leader and follower. The leader (*dashed box*) is denoted by y_l and x_l, while the follower gets y_f and x_f. Note that the y-axis is situated in the longitudinal direction (along the direction of movement) and the x-axis in the lateral direction. Angle β, or the absolute orientation of the follower with respect to the leader, marks the rotation between the two coordinate frames

In the context of this study, we take into account three mass points: (m_M) at the center, (m_L and m_R) at the left and right tread respectively (Fig. 15.6). In this figure, B represents the width of the robot, F_{fric} is the friction force and T_{fric} is the torque caused by the friction force. F_L and F_R and the forces provided by respectively the left and right track. Three so-called generalized coordinates constitute the degrees of freedom to the system: x_M, y_M and β. We work in a coordinate frame attached to the leader.

Based on the generalized coordinates, we derive an expression for the kinetic and potential energy of the system, after which the Lagrangian L is defined by taking the difference between the two types of energy. The system's dynamic equations are obtained in the following manner:

$$\frac{\partial L}{\partial q_j} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_j}\right) + Q_{j\,r.n.c.} = 0; \quad j = 1, 2, 3.$$

Here, q_j stands for the generalized coordinates, while $Q_{j\,r.n.c.}$ indicates the real non-conservative forces in the respective j-direction. Generalized coordinates in each j-direction $\{q_1, q_2, q_3\}$ correspond respectively to $\{x_M, y_M, \beta\}$. In our model, the forces are exerted by the DC motors on the treads, frictional forces and torque. Moreover, we presume them to be proportional to the speed in the corresponding direction, selecting C_i as proportional constants. The force is considered to be proportional to the voltage supplied to the motors as its dependence on the speed is limited. By doing this, we not only make sure that the equations remain clear and easy to handle, we also obtain the following equations which are valid in a reference frame fixed on the leader:

$$m_{tot}\ddot{x}_M = \sin(\beta)\left(V_L\frac{K_\Phi}{R_a r_L} + V_R\frac{K_\Phi}{R_a r_R}\right) - C_1\dot{x}_M \tag{15.1}$$

$$m_{tot}\ddot{y}_M = \cos(\beta)\left(V_L\frac{K_\Phi}{R_a r_L} + V_R\frac{K_\Phi}{R_a r_R}\right) - C_2\dot{y}_M \tag{15.2}$$

$$mB\ddot{\beta} = \frac{K_\Phi}{R_a r_R}V_R - \frac{K_\Phi}{R_a r_L}V_L - \frac{2C_3}{B}\dot{\beta} \tag{15.3}$$

where:

- r_L and r_R denote the radius of the left and right treads (ideally the same and assumed in the remainder of this chapter)
- m is the mass of one track (It is assumed that the left and right track have the same mass, $m = m_L = m_R$)
- m_{tot} is the total mass including the mass of the two tracks and the mass of the central part of the robot ($m_{tot} = m_M + m_L + m_R$)
- B is the width of the robot as can be seen in Fig. 15.6
- V_L and V_R denote the voltage going to the left and right DC motors
- R_a is the resistance of the armature winding
- K_Φ is the flux constant of the DC motors

On the left-hand side of equations (15.1)–(15.3) we see the three controlled variables: x_M, y_M and β. The two manipulated variables are on the right-hand side, illustrating the voltages going to the motors (V_L and V_R).

In order to design controllers for this system, transfer functions are needed which are obtained from system identification. First, we identify the β-equation, based on the Laplace transform of equation (15.3). When choosing the orientation via the voltage going to the right motor (i.e. $V_L = 0$), we obtain the following identified equation using the step response:

$$\mathcal{L}(\dot{\beta}(t)) = s\beta(s) = s\frac{0.81}{0.22s^2 + s}V(s).$$

where $\mathcal{L}$ represents the Laplace transform, $\beta(s)$ and $V(s)$ represent the Laplace transformations of $\beta(t)$ and $V(t)$, β is expressed in $^\circ$ and V is expressed in volt.

Again, to tune the required longitudinal controller, Eq. (15.2) has to be identified. Therefore, we linearize the equation around $\beta = \ddot{\beta} = \dot{\beta} = 0$, $y = y_{ref}$ and $\ddot{y}_M = \dot{y}_M = 0$. After transforming to the Laplace domain, we obtain the following identified equation using the step response:

$$\mathcal{L}(\dot{Y_M}(t)) = sY_M(s) = s\frac{-0.997}{0.203s^2 + s}V(s).$$

where $Y_M(s)$ and $V(s)$ represent the Laplace transformations of $Y_M(t)$ and $V(t)$.

As can be seen in Sect. 15.3.2, the controller for the x_M direction will be tuned using the an autotuning technique which makes explicit identification of the x_M equation unnecessary.

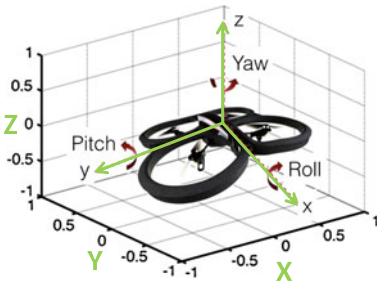


Fig. 15.7 Three axes of motion for the quadrotor and the rotations around them: Pitch, Yaw and Roll

Aerial Agent

The quadrotor aerial movements are similar to those of a conventional helicopter. The difference is that movement is achieved by varying each of the motor speeds to obtain the desired effect. Figure 15.7 depicts the movement axes of the quadrotor. The 4 Degrees Of Freedom (DOF) of the AR.Drone permit changes in altitude and position. Movements are thus achieved on:

- Pitch—By rotational movement along transverse axis y, translational movement on x axis is made.
- Roll—By rotational movement along longitudinal axis x, translational movement on y axis is made.
- Yaw—Rotational movement along z axis.
- Throttle—Translational movement on z axis.

The control parameters given to the internal controllers are floating point values between [–1,1] and represent the percentage of the minimum or maximum configured value for the respective movement. For the yaw only an angular speed can be send to the internal controller. We denote $\{\theta, \phi, \psi, v_z, v_\psi\}$ the pitch angle reference, the roll angle reference value, the yaw angle reference, the vertical speed reference, and the yaw angular speed reference. The translational displacements in all three directions are denoted as $\{x, y, z\}$. The roll, pitch and yaw angles are given in radians, while altitude is expressed in meters. Translational speed is expressed in m/s and angular speed in rad/s. Due to the low layer internal control, the quadrotor behaves as a Linear Time-Invariant System. Making it possible to perform for each degree of freedom a parametric identification using the prediction error method (Ljung 2007). A Pseudo-Random Binary Signal (PRBS) is used to identify the dynamics of the quadrotor. A sampling time of 66 ms is chosen based on the analysis of dynamics performed on previous work (Vlas et al. 2013; Hernandez et al. 2014b).

The obtained transfer functions are given by:

$$H_x(s) = \frac{x(s)}{\theta(s)} = \frac{7.27}{s(1.05s+1)}$$
$$H_y(s) = \frac{y(s)}{\phi(s)} = \frac{7.27}{s(1.05s+1)}$$

$$H_z(s) = \frac{z(s)}{v_z(s)} = \frac{0.72}{s(0.23s+1)}$$
$$H_{yaw}(s) = \frac{\psi(s)}{v_\psi(s)} = \frac{2.94}{s(0.031s+1)}$$

15.3.2 Low-Level PID Cascade Control

The low-level PID cascade control has the purpose to control the position of the followers with respect to the leader. The distance between leader and follower is obtained from image processing.

Image Processing

To calculate the position and orientation of the followers in relation to their leader, each of them continuously takes pictures of its leader's marker and executes an image processing algorithm. The markers themselves are rectangular in shape, having a height (H) of 4 cm and a length (L) of 8 cm. To simplify the calculations, the markers are put in such way that the upper side of the rectangles is at the same height as the camera, which is then perpendicular to the top of the markers. On the camera we locate a coordinate frame, in which the y-coordinate denotes the distance along the camera, the x-coordinate the distance perpendicular to the camera and the z-coordinate the height relative to the camera. Figure 15.8 shows the configuration. The goal is to determine the x_M- and y_M-coordinate, together with the orientation β of the follower with respect to the leader. Note that these are the three controlled variables mentioned above. The first two are given by (15.4) and (15.5). To calculate β, we first need to calculate θ which is the orientation of the leader in relation to the plane parallel to the camera (Fig. 15.8):

$$x_M = \frac{x_A + x_C}{2}; \tag{15.4}$$
$$y_M = \frac{y_A + y_C}{2}; \tag{15.5}$$

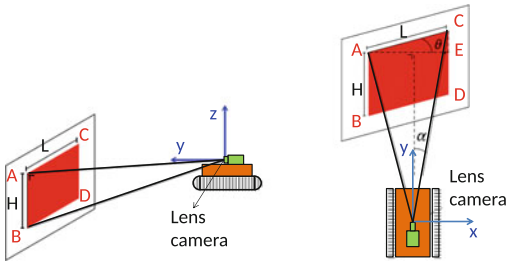


Fig. 15.8 Side and top view of the configuration

$$\theta = cos^{-1}\left(\frac{x_A - x_C}{L}\right). \tag{15.6}$$

After taking a picture, the search algorithm looks for rectangular areas of which the pixels match a predefined color interval. It then stores the coordinates of the four corners, expressed in pixels. In order for the pixel values to be converted in cm, a lookup table is made linking the amount of pixels in the image for the height of the marker to the corresponding value in cm for a certain distance between the camera lens and the marker. This step is an image calibration step.

The angle θ in equation (15.6) is the orientation of the leader in relation to the plane parallel to the camera. In other words, the orientation of the leader in the reference frame attached to the camera. However, as we need the absolute orientation of the follower with respect to the leader, i.e. in a frame fixed to the follower, we execute the following equation:

$$\beta = \text{orientation} = \theta - \alpha.$$

To calculate these coordinates, it is necessary for the camera to always have a clear sight of one of the leader's markers. In view of this goal, we designed an additional control strategy for the camera so that it automatically follows the marker.

Control Strategy

Camera Control

A control strategy is designed to make sure the camera always directs itself towards the center of one of the markers (Fig. 15.9). This strategy consists of a proportional controller that adapts its P-term depending on the longitudinal distance of the follower with respect to the leader. Depending on this longitudinal distance one step of the stepper motor corresponds with a different pixel displacement. We obtain the adaptation of the P-term by means of a table that is the result of an experiment recording the pixel displacement corresponding to one step of the stepper motor at different longitudinal distances. Each time a picture is taken, the required number of steps is calculated by taking the difference between the middle of the picture and the middle of the marker. To counter the mechanical inertia of the stepper motor, a delay of 85 ms is built in whenever the camera rotates. The stepper motor can reach its goal position before a new picture is taken. Since the image processing algorithm requires about 200 ms to be executed, three to five updates per second are possible. When a follower sees different markers, only the one situated closest is taken into

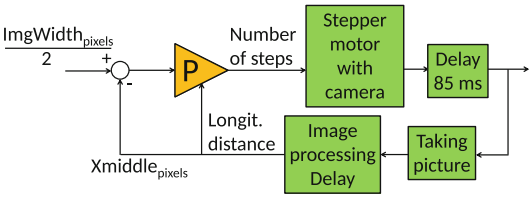


Fig. 15.9 Control strategy to position the camera always at the *center* of the marker of the leader

consideration as this is the one most likely to be its leader. The angle α necessary for the calculate β is known because every step of the stepper motor is counted,

Vehicle Control

Based on equations (15.1)–(15.3), we developed a control strategy that entails some considerations. Firstly, we want the absolute orientation angle β to be equal zero, because then the follower and leader are oriented in the same direction, which is necessary to maintain the formation. Since (15.3) is a linear equation, it is easy to manipulate and build a controller. Secondly, when we manage to obtain the desired value for the absolute orientation of the follower, the x-coordinate can no longer be altered by varying V_L and V_R because of $\sin(\beta) = 0$. The y-coordinate, by contrast, can still be altered by adapting the exact same values V_L and V_R. Note that to adapt the y-coordinate, there is no need to change β or the x-coordinate.

After taking these items into consideration, a cascade control strategy controlling both the x-coordinate and the orientation β can be chosen. The correlation between x-coordinate and orientation underlies this control strategy, given that neither can be changed without affecting the other. Within the strategy, the primary control variable is controlled by manipulating the set-point of a secondary control variable. As such, it consists of a loop within a loop (Fig. 15.10).

Since the orientation can be adapted quickly, we chose it as our secondary variable. The x-coordinate, by contrast, changes rather slowly, making it an ideal primary variable. In order for cascade control to be successful, it is necessary that the outer loop is at least four times slower than the inner loop (based on practical insight of applications). Hence, we strongly limited the control action of the primary controller to artificially slow down the outer loop.

First, the PI used to control the orientation β is designed by using FRTool (De Keyser and Ionescu 2006), with the following design specifications:

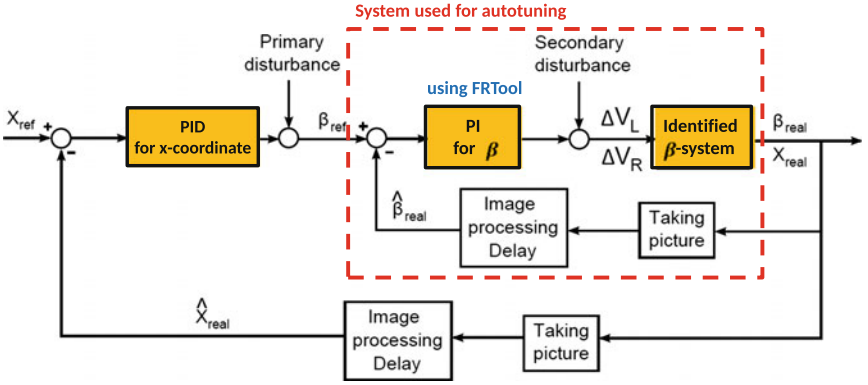


Fig. 15.10 Cascade control strategy for x-coordinate and orientation β. $\hat{\beta}_{real}$ and $\hat{x}_{real}$ represent the value for β and x measured using image processing techniques

robustness > 0.75, settling time $< 4\,\text{s}$ and overshoot $< 20\,\%$. The transfer function of the resulting controller is:

$$\frac{U_\beta(s)}{\beta_{ref}(s) - \beta(s)} = C_\beta(s) = \frac{1.6(s + 0.4)}{s}.$$

Secondly, the outer loop has to be tuned to complete the cascade control strategy. In fact, adding the inner controller causes the loop to have an order higher than 2. Apart from this, there is no information coming from the wheel encoders about the speed in the x-direction. To solve these problems we used the auto-tuning method of Åström-Hägglund (Aström and Hägglund 2006) to build a primary controller as seen in (Ionescu and De Keyser 2012) with the relay-value (d) being 7°. The outcome of the experiment (measured output as well as the fitted curve) is shown in Fig. 15.11. The resulting PID parameters are: $\text{K}_{px} = 0.89$, $\text{T}_{ix} = 3.50$ and $\text{T}_{dx} = 0.87$.

Regarding the y-coordinate, we choose a simple feedback control strategy to control the longitudinal distance between the leader and follower (Fig. 15.12). As already mentioned, this is possible because we are able to adapt the y-coordinate without changing the the x-coordinate or the orientation.

The PI used to control the y-coordinate is designed using FRTool (De Keyser and Ionescu 2006), with the following design specifications: robustness > 0.75, settling time < 3 s and overshoot $< 15\,\%$. The transfer function of the resulting controller is:

$$\frac{U_y(s)}{Y_{ref}(s) - Y(s)} = C_y(s) = \frac{-2(s + 0.274)}{s}.$$

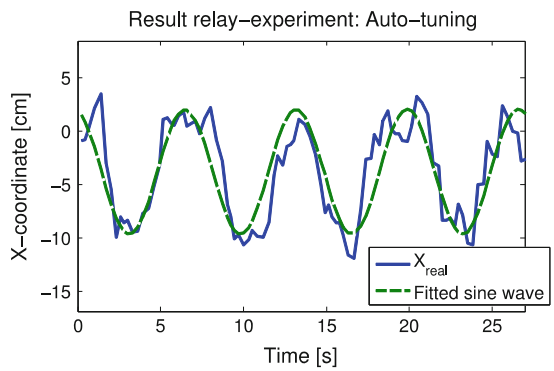


Fig. 15.11 Result of the performed relay experiment

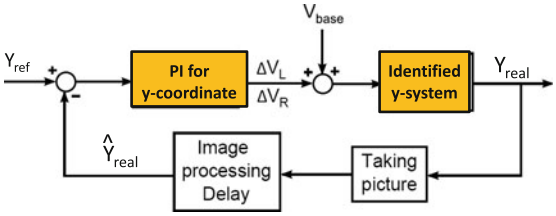


Fig. 15.12 Control strategy for y-coordinate. $\hat{Y}_{real}$ represents the value for Y measured using image processing techniques

15.3.3 High-Level Model-Based Predictive Control

This section describes the path-following control strategy for the quadrotor. This controller will send the setpoints to the black-box internal controller of the AR.Drone. It is important to notice that due to this internal control, each degree of freedom in the quadrotor behaves independently, thus making it possible to tune SISO controllers. First the general Model-based Predictive Control strategy is discussed. Afterwards, an explanation is given to obtain reference points for the controller based on image processing techniques.

Model-Based Predictive Control

Model-based Predictive Control (MPC) is a general designation for controllers that make explicit use of a model of the plant to obtain the control signal by minimizing an objective function over a time horizon in the future. In this contribution we will make use of the Extended Prediction Self-Adaptive Control (EPSAC) approach to MPC. This methodology proposed by (De Keyser 2003) is briefly described in the following paragraphs.

The generic model of the EPSAC algorithm is:

$$y(t) = x(t) + n(t) \tag{15.7}$$

where $y(t)$ is the measured output of the process, $x(t)$ is the model output and $n(t)$ represents model/process disturbance, all at discrete-time index t. The model output $x(t)$ represents the effect of the control input $u(t)$ on the process output $y(t)$. It can be described by the following equation:

$$x(t) = f\,[x(t-1), x(t-2), \ldots, u(t-1), u(t-2), \ldots]$$

Notice that $x(t)$ represents here the model output, not the state vector. Also important is the fact that f can be either a linear or a nonlinear function.

The disturbance $n(t)$ can be modeled as colored noise through a filter with the transfer function

$$n(t) = \frac{C(q^{-1})}{D(q^{-1})} e(t)$$

with $e(t)$ uncorrelated (white) noise with zero-mean and C, D monic polynomials in the backward shift operator q^{-1}. The disturbance model allows to achieve robustness of the control loop against unmeasured disturbances and model errors. A possible way to remove steady-state control offsets is $n(t) = \frac{1}{1-q^{-1}} e(t)$ (Maciejowski 2002).

A fundamental step in the MPC methodology is the prediction. Using the generic process model (15.7), the predicted values of the output are described by

$$y(t+k|t) = x(t+k|t) + n(t+k|t)$$

for $k = N_1, N_1 + 1, \ldots, N_2 | N_1, N_2 \in \Re$, where N_1 and N_2 are the minimum and the maximum prediction horizons. The prediction of the process output is based on the measurements available at the sampling instant t, $\{y(t), y(t-1), \ldots, u(t-1), u(t-2), \ldots\}$ and future (postulated) values of the input signal $\{u(t|t), u(t+1|t), \ldots\}$. The future response can then be expressed as

$$y(t+k|t) = y_{base}(t+k|t) + y_{opt}(t+k|t), \tag{15.8}$$

where each of the contribution terms is understood as:

- $y_{base}(t+k|t)$ is the effect of the past inputs $u(t-1), u(t-2) \ldots$, a future base control sequence $u_{base}(t+k|t)$ that can be the last used input and the predicted disturbance $n(t+k|t)$.

- $y_{opt}(t+k|t)$ is the effect of the optimizing control actions $\delta u(t|t), \ldots, \delta u(t+N_u-1|t)$ with $\delta u(t+k|t) = u(t+k|t) - u_{base}(t+k|t)$, in a control horizon N_u.

The optimized output $y_{opt}(k) \, \forall \, k = [1, 2, \ldots, N_2]$ can be expressed as the discrete time convolution of the unit impulse response coefficients $h_1, \ldots, h_{N_2}$ and unit step response coefficients $g_1, \ldots, g_{N_2}$ of the system as.

$$\begin{aligned} y_{opt}(t+k|t) = {} & h_k \delta u(t|t) + h_{k-1} \delta u(t+1|t) + \ldots \\ & + g_{k-N_u+1} \delta u(t+N_u-1|t) \end{aligned} \tag{15.9}$$

Using (15.8) and (15.9), the key EPSAC formulation becomes

$$\mathbf{Y} = \overline{\mathbf{Y}} + \mathbf{G}\mathbf{U}$$

where

$$\begin{aligned} \mathbf{Y} &= \left[y(t+N_1|t) \ldots y(t+N_2|t) \right]^T \\ \overline{\mathbf{Y}} &= \left[y_{base}(t+N_1|t) \ldots y_{base}(t+N_2|t) \right]^T \\ \mathbf{U} &= \left[\delta u(t|t) \ldots \delta u(t+N_u-1|t) \right]^T \end{aligned}$$

$$\mathbf{G} = \begin{bmatrix} h_{N_1} & h_{N_1-1} & \ldots & g_{N_1-N_u+1} \\ h_{N_1+1} & h_{N_1} & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ h_{N_2} & h_{N_2-1} & \ldots & g_{N_2-N_u+1} \end{bmatrix}$$

Then, the control signal U is optimized by minimizing the cost function:

$$J = \sum_{k=N_1}^{N_2} \left[r(t+k|t) - y(t+k|t) \right]^2 \tag{15.10}$$

Note that the controller cost function (15.10) can be easily extended to many alternative cost functions (similar to the approach in optimal control theory) as described in (De Keyser 2003). The horizons N_1, N_2 and N_u are design parameters and $r(t+k|t)$ is the desired *reference trajectory*.

The cost function (15.10) can be represented in its compact matrix notation as follows:

$$(\mathbf{R}-\mathbf{Y})^{\mathbf{T}}(\mathbf{R}-\mathbf{Y}) = [(\mathbf{R}-\overline{\mathbf{Y}})-\mathbf{GU}]^{\mathbf{T}}[(\mathbf{R}-\overline{\mathbf{Y}})-\mathbf{GU}]$$

where $\mathbf{R} = [r(t+N_1|t)\ldots r(t+N_2|t)]^T$.

The previous expression can be easily transformed into the standard quadratic cost index:

$$J(\mathbf{U}) = \mathbf{U}^{\mathbf{T}}\mathbf{H}\mathbf{U} + \mathbf{2f}\mathbf{U} + c. \tag{15.11}$$

with,

$$\mathbf{H} = \mathbf{G}^{\mathbf{T}}\mathbf{G} \quad \mathbf{f} = -\mathbf{G}^{\mathbf{T}}(\mathbf{R}-\overline{\mathbf{Y}})$$
$$c = (\mathbf{R}-\overline{\mathbf{Y}})^{\mathbf{T}}(\mathbf{R}-\overline{\mathbf{Y}})$$

where $[G^T G] \quad \in \Re^{N_u \times N_u}$. The solution of minimizing (15.11) is:

$$\mathbf{U}^* = [\mathbf{G}^{\mathbf{T}}\mathbf{G}]^{-\mathbf{1}}[\mathbf{G}^{\mathbf{T}}(\mathbf{R}-\overline{\mathbf{Y}})]$$

Finally, the feedback characteristic of MPC is given as the first optimal control input $u^*(t) = u_{base}(t|t) + \delta u(t|t) = u_{base}(t|t) + U^*(1)$ is applied to the plant and then the whole procedure is repeated again at the next sampling instant $t+1$.

The EPSAC strategy is implemented for all degrees of freedom. The main specification was to achieve a fast response without overshoot. A combination of long prediction horizon (N_2) and short control horizon (N_u) was also considered in order to introduce a higher robustness in the controller (De Keyser 2003). The tuned EPSAC parameters are summarized in Table 15.1. Note that $N_1 = 1$ as there is no time delay in the system.

As comparison for the performance of the EPSAC strategy, also a PD controller was designed for the quadrotor. The PD controller was tuned using FRtool (De Keyser and Ionescu 2006) as it provides an intuitive graphical interface based on the Nichols plot (James et al. 1965) using design specifications such as: robustness, settling time,

Table 15.1 EPSAC controller parameters

SISO system	N_1	N_2	N_u	Noise Filter: C/D
x, y	1	15	1	$\frac{1}{1-q^{-1}}$
z	1	30	1	
yaw	1	10	1	

phase margin and/or gain margin. The PD controller is represented in (15.12), where: N is the derivative gain limit; usually an integer number. PD controllers with filter action (15.12) can be represented as one gain, one zero and one pole (15.13).

$$C_{(s)} = K_p + K_p \frac{T_d s}{\frac{T_d}{N} s + 1} \tag{15.12}$$

$$C_{(s)} = K_p(N+1) \frac{s + \frac{N}{T_d(N+1)}}{s + \frac{N}{T_d}} \tag{15.13}$$

Once the controller is exported from FRTool in zero-pole-gain form (15.14):

$$C_{(s)} = K \frac{(s + z_1)}{(s + p_1)} \tag{15.14}$$

The PD parameters K_p,T_d and N can be calculated by defining a system of equations from (15.13) and (15.14).

$$\begin{aligned} K &= K_p(N+1) \\ z_1 &= \frac{N}{T_d(N+1)} \\ p_1 &= \frac{N}{T_d} \end{aligned}$$

The design specifications: robustness (Ro), settling time (T_{set}), overshoot percent ($\%OS$) and gains of the tuned PD controllers are presented in Table 15.2.

Reference Setpoints Obtained from Image Processing

The goal of the quadrotor position control is to obtain a situation where the quadrotor automatically follows the ground agents. The reference setpoints are obtained using image processing based on the images captured with the bottom camera of the quadrotor as suggested in (Hernandez et al. 2014a). The ground agents are recognized by the quadrotor using pattern recognition. This process consists in delineating the black and white image with an ordered sequence of pixels. Subsequently, the contours can be recognized as polygonal approximations by identifying only the squares using an exclusion analysis (for example the number of polygon sides).

Once the robots are identified, the angle and distance between robots is computed. The procedure consists of 5 steps. First, the center of the image (CI) is computed, this

Table 15.2 Design parameters for the PD controllers

Controller	Ro	T_{set}	$\%OS$	K_p	T_d	N
x,y	≥ 0.7	$\leq$5 s	$\leq$3 %	0.15	0.96	1
z	$\geq$0.7	$\leq$5 s	$\leq$3 %	1.6	0.46	1
ψ	$\geq$0.7	$\leq$5 s	$\leq$3 %	1.52	0.08	1

basically represents the position of the quadrotor above the robots. Note that the real position of the quadrotor respectively to the UGVs must be corrected depending on the current pitch angle ϕ and roll. The second step consists in calculating the center of the square that represents the ground vehicle to be followed (CR).

Next in the third step, the error between the position of the quadrotor (CI) and the position of the ground robot (CR) is calculated and converted from pixels to meters. Considering that the center of the ground vehicle has a coordinate $P_{ci} = (x_{ci}, y_{ci})$ and the orientation point $P_{oi} = (x_{oi}, y_{oi})$, where the sub-indexes c refers to center and o to orientation of the ith ground robot. The conversion factor from pixels to meters is easily found because the resolution of the camera and the real distance between P_{ci} and P_{oi} are known for a given altitude (Fig. 15.13a).

In the fourth step, the orientation φ of the ground robot with respect to the quadrotor is computed (Fig. 15.13b) as:

$$\varphi = tan^{-1}\left(\frac{x_{c1} - x_{o1}}{-(y_{c1} - y_{o1})}\right)$$

where tan^{-1} represents the 'atan2' function which results in angles defined in all four quadrants.

The final step consists in computing the lateral and longitudinal distances which will be later given to the controller of the ground agents. In the case that the formation is aligned with respect to the orientation of the quadrotor as in Fig. 15.14a, the

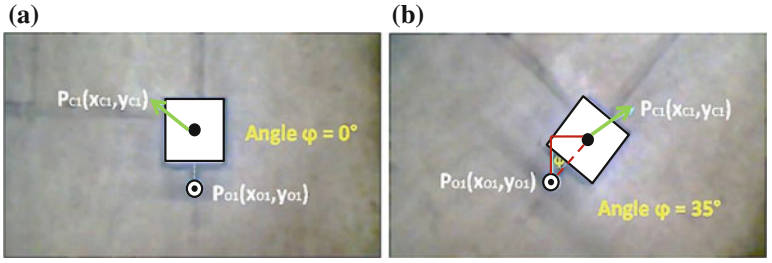


Fig. 15.13 Orientation of the quadrotor with respect to the ground vehicle

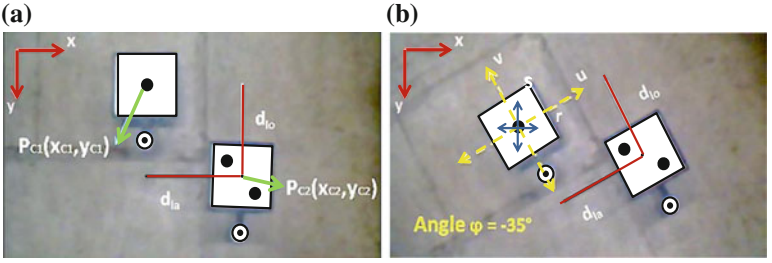


Fig. 15.14 Calculation of lateral and longitudinal distances

computation of the lateral distance d_{la} and longitudinal distance d_{lo} of the followers can be obtained directly from the points representing center of the robots $P_{ci} = (x_{ci}, y_{ci})$ for $i = 1, 2, \ldots$ representing the subindex for each ground agent.

$$d_{lo} = y_{c2} - y_{c1}$$
$$d_{la} = x_{c2} - x_{c1}$$

However, for the case when the formation is not aligned with respect to the quadrotor orientation as shown in Fig. 15.14b, a change of coordinates is required in order to correctly compute the distances. First, a new Cartesian plane (r, s) is created with coordinate $(0, 0)$ in the center of the first robot, leading to the following new P_1 and P_2 points:

$$\begin{aligned} P_{c1}(r, s) &= (0, 0) \\ P_{c2}(r, s) &= ((x_{c2} - x_{c1}), -(y_{c2} - y_{c1})) \end{aligned} \tag{15.15}$$

Subsequently, $P_1(r, s)$ and $P_2(r, s)$ in (15.15) are transformed into polar coordinates in the plane (r, s):

$$P_{c2}(r, s) = \left(\sqrt{\Delta X^2 + \Delta Y^2}, tan^{-1}\left(\frac{-\Delta Y}{\Delta X}\right)\right)$$

where $\Delta X = x_{c2} - x_{c1}$ and $\Delta Y = y_{c2} - y_{c1}$. Then $P_2(r, s)$ is rotated by φ and mapped to the plane (u, v):

$$\begin{aligned} P_{c1}(u, v) &= (0, 0) \\ P_{c2}(u, v) &= \left(\sqrt{\Delta X^2 + \Delta Y^2}, tan^{-1}\left(\frac{-\Delta Y}{\Delta X} + \varphi\right)\right) \end{aligned} \tag{15.16}$$

Finally, the lateral and longitudinal distances d_{la} and d_{lo} are obtained from (15.16):

$$d_{la} = \sqrt{\Delta X^2 + \Delta Y^2}\, cos\left(tan^{-1}\left(\frac{-\Delta Y}{\Delta X} + \varphi\right)\right)$$
$$d_{lo} = \sqrt{\Delta X^2 + \Delta Y^2}\, sin\left(tan^{-1}\left(\frac{-\Delta Y}{\Delta X} + \varphi\right)\right)$$

15.4 Experimental Results

The application in this research is from precision agriculture (PA). A series of experiments is performed to serve as a proof-of-concept for a MAS used in PA where robots are used to observe and measure inter- and intra-field variability. In a planted

field it is of utmost importance that growth conditions are ubiquitously optimal. To observe and measure growth conditions in large field areas, a group of UGVs can be used to perform this task as quickly as possible. However, they have restricted movement as they need to avoid the crop lines in order not to damage the crops. For this purpose, formation control of the UGVs is necessary. The quadrotor acts as a flying sensor in order to provide the UGVs with additional information. Therefore, he needs to be able to recognize the UGV's and follow them.

A first experiment is performed to illustrate the formation control of the UGVs. In this experiment the formation is fixed according to the width of the crop lines. The second experiment shows the performance of the model predictive controller for the quadrotor. A third experiment is performed to see whether the quadrotor is able to recognize and follow the leader of the UGVs and give reference signals to the followers.

15.4.1 Formation Control of UGVs

The formations used in this experiment and the *in-line* and *triangular* formations described in Sect. 15.2.

To test the designed and implemented controllers, we let the leader drive in a straight line for about 24 s, take a 90° bend to the right, and finally drive in a straight line for about 10 s. Within this trajectory, the bend represents a disturbance testing the control strategy on its ability to reject such perturbations. The followers drive in a triangular formation, to an angle of ±30° and a y-coordinate of 30 cm, corresponding to an x-coordinate of ±18 cm, in relation to the leading robot. The monitored x- and y-coordinate for both the left and right followers are shown in Fig. 15.15. The corresponding control action can be seen in Fig. 15.16.

Figure 15.15 illustrates that the y-controller manages to control the y-coordinate accurately during the straight line, as well as in the bend. The x-coordinate, by contrast, is more accurately controlled during the straight line. In the bend, both

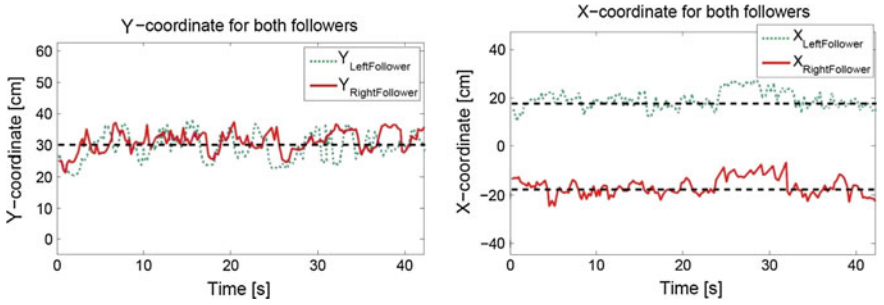


Fig. 15.15 $y(t)$ and $x(t)$ for both followers

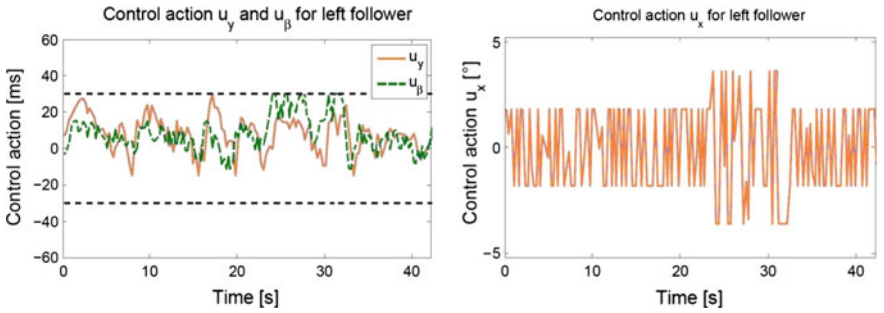


Fig. 15.16 Control action: u_y(t), u_β(t) and u_x(t) for left follower

followers deviate from their paths with an offset in the lateral coordinate as a result. However, this deviation can be rapidly altered, causing the followers to reach the desired formation quite fast. In order for this offset to disappear in the bend, an extra integrator can be added to the control strategy. Figure 15.16 illustrates that u_β would only exceed its allowed interval of [–30,…,30] ms in the bend, whereas u_y would not. In order to fulfill the cascade criterion, u_x is limited to the interval of [–1.8,…,1.8°]. Yet, to make sure it reacts more accurately in case of a disturbance, it is allowed to range between [–3.6,…,3.6°] in the bend.

The designed controllers are able to control both followers accurately. Taking into account that no two robots are identical, we can state that the designed controllers are sufficiently robust.

Finally, as this experiment indicates, it is clear that the control actions u_y and u_β correlate, i.e. their evolution in time is similar. This implies that it might be possible to work with two independently controlled variables instead of three, which would reduce the number of degrees of freedom in the system and make the control strategy easier.

15.4.2 Path Following for the Quadrotor

In this subsection a path following experiment is performed to test the capabilities of the controller of the quadrotor to follow a setpoint. Special attention is paid to the settling time and overshoot, as this will limit the control performance for path-following. A faster controller allows the quadrotor to perform more aggressive maneuvers, whilst a smaller overshoot allows it to more accurately follow the trajectory in confined spaces.

The performance of both the MPC and the PD controller is shown in Fig. 15.17, where the quadrotor is requested to follow four setpoints in the 3D space. The task consists in sequentially following the waypoints, starting at point 0. The EPSAC

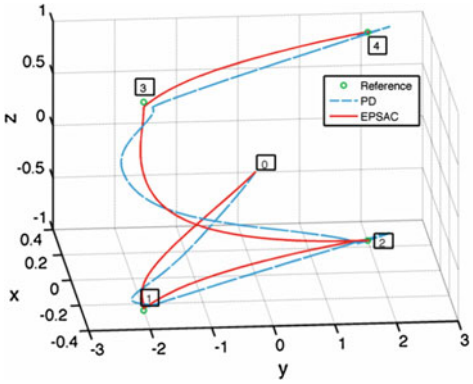


Fig. 15.17 3D response for path-following of the AR.Drone 2.0. The markers represent the sequential waypoints for the quadrotor in the space

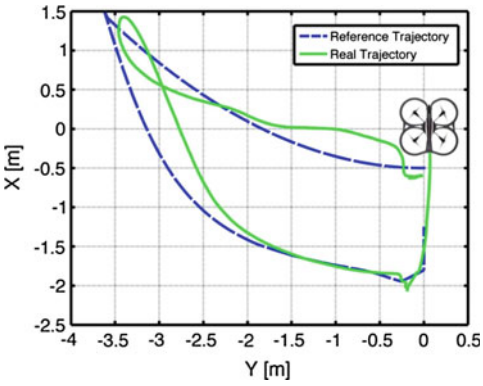


Fig. 15.18 Results obtained for the PD controllers following a trajectory in a two dimensional space

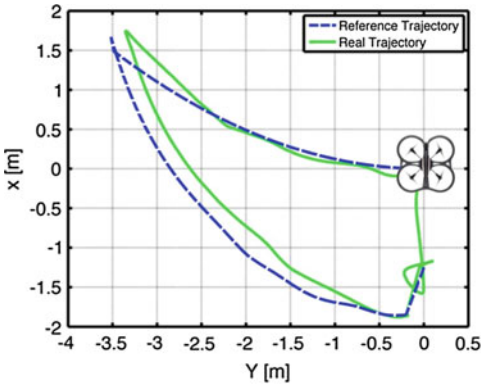


Fig. 15.19 Results obtained for the EPSAC controllers following a trajectory in a two dimensional space

is able to follow more accurately the path without large deviations. The 2D-representation for this experiment is depicted in Fig. 15.18 for the case of the PD controllers. Observe the large oscillations of the quadrotor around its reference trajectory. In Fig. 15.19 the same is shown for the case of the MPC (designed in previous

Table 15.3 Performance index for Path-following strategies

Controller	ISE	IAE	ITAE
PD X	118.12	159.99	1862.7
EPSAC X	**100.81**	**134.2**	**1279.6**
PD Y	87.28	164.65	2360.5
EPSAC Y	**68.89**	**131.68**	**1990.9**

section). Observe that the path of the quadrotor is less oscillatory compared to the PD controller.

In the case of the EPSAC path-following controller, the quadrotor experiences less deviations to the desired trajectory, specially at the moment of performing sharp bends as in the middle of the trajectory. The performance of the controllers is further compared using the well-known performance index Integral Squared Error (ISE), Integral Absolute Error (IAE), and Integral Time-weighted Absolute Error (ITAE) (Soni and Bhatt 2011); which are used in this example to compute the error between the reference and the real trajectory described by the quadrotor. As expected the errors are less for the proposed MPC strategy as summarized in Table 15.3.

15.4.3 Quadrotor as Flying Sensor for Ground Agents

In this subsection, an experiment is performed where the quadrotor follows the ground agents automatically. The experiment consists in keeping a triangular formation of three robots, one leader and two followers by using the control algorithm described in previous sections. The area covered by the triangular formation of the UGVs varies depending on the width of the crop lines. As this area increases the quadrotor should automatically alter its height to get the total formation in the field-of-view.

The quadrotor uses image processing to follow the ground agents as discussed in Sect. 15.3.3. The final result of the image processing is presented in Fig. 15.20, which includes the following information: the calculation of distances, the simulation time (on the bottom-left), the altitude obtained from the ultrasonic sensor and the one based on the image (at the bottom-right), the leader robot being followed (at top-left), the orientation (φ in degree) of each robot with respect to the quadrotor and the lateral and longitudinal distances are represented by the black lines with origin in the center of each follower robot.

For the described experiment, the obtained performance for the triangular formation is seen in Figs. 15.21 and 15.22. Note that the distance is controlled without steady state error but with a long settling time. It is possible to observe that the longitudinal and lateral distances are controlled after a big change in the setpoint ($\pm$20 cm at time 14 and 26 s for couple 1–2 and at time 22 and 34 s for couple 1–3) without steady-state error. The robustness of the controller designed is also tested during the experiment, as the same controller (i.e. longitudinal and lateral) is used for

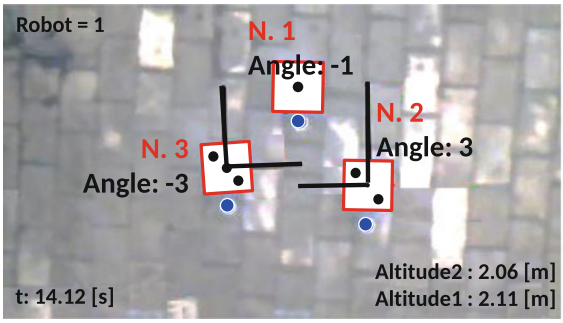
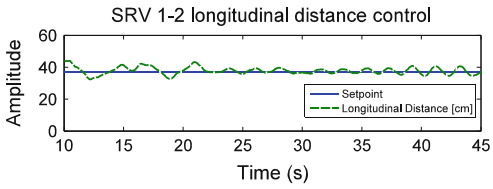


Fig. 15.20 Screenshot of the result obtained after the image processing



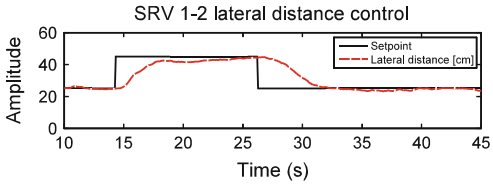
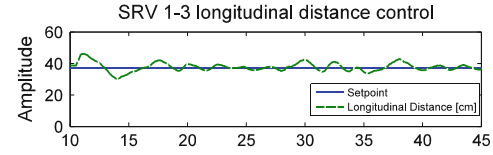


Fig. 15.21 Results of formation control for robot couple 1–2



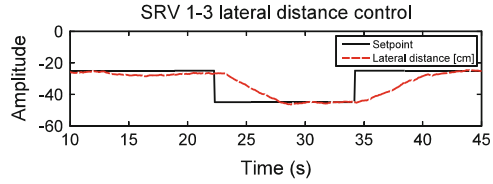


Fig. 15.22 Results of formation control for robot couple 1–3

all the robots, despite the robots presenting quite different dynamics which moreover change on time due to errors in the encoder, bad contacts or wear on the gear-box.

The performance of the position control for the quadrotor is shown in Fig. 15.23. The quadrotor follows the position of the ground leader precise and smoothly. The altitude of the quadrotor is 2 m (Fig. 15.23a) and because the formation remained on a straight formation the orientation was always zero (Fig. 15.23b). Finally, the setpoint

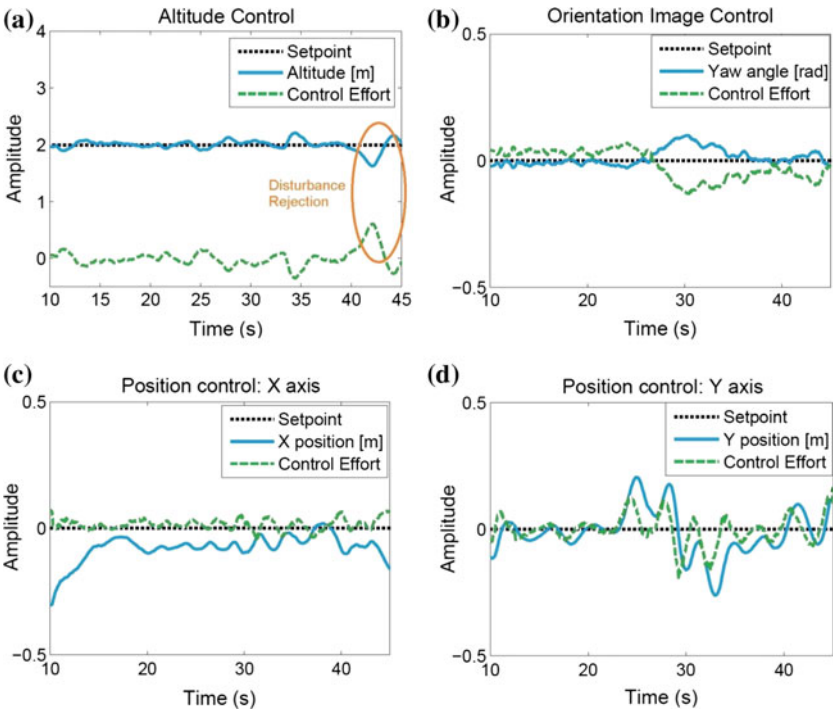


Fig. 15.23 Position control of the quadrotor during formation control. **a** Altitude. **b** Orientation. **c** X position. **d** Y position

for the X- and Y-movements of the quadrotor is zero, because it is considered as the error between the quadrotor and the ground leader.

15.5 Conclusions

This study describes a multi agent system (MAS) for precision agriculture but which can also be easily applied to traffic control, search and rescue, etc. The MAS consists of a group of unmanned ground vehicles (UGVs) and an unmanned aerial vehicle (UAV) which acts as flying sensor to assist the UGVs to observe, measure inter- and intra-field variability in crops. A PID cascade controller strategy for the UGV leader-follower system is designed based on a derived mathematical model. The path-following task of the quadrotor is performed using a model-based predictive control strategy (EPSAC). The developed controllers are tested in a series of experiments. To use the quadrotor as a flying sensor, a third experiment is performed where the quadrotor follows the ground agents using image processing and the designed position controller. The results suggest a good control performance but future robustness studies will be needed. Future work also includes an experiment where the quad rotor

is sending reference signals to the ground agents. In this way obstacle avoidance for the ground agents can be obtained and a fast response can be given to changing field conditions such as dehydration which the quadrotor can detect based on image processing.

Acknowledgments Clara M. Ionescu is a post-doc fellow of the Research Foundation—Flanders (FWO). The authors acknowledge the strategic research center Flanders Make.

References

Aström KJ, Hägglund T (2006) Advanced PID control. Instrument Society of America, North Carolina NC

Beard C, Chen Z, Kumar V, Lee Y, Leon-Salas W, Rao P (2013) Saveus: Saving victims in earth quakes through unified systems. Int J Commun Netw Distrib Syst 10(4):402–420

Bristeau PJ, Callou F, Vissiere D, Petit N (2011) The navigation and control technology inside the AR. Drone micro UAV. In: Proceedings of 18th IFAC world congress, Milano, Italy, 28 August-2 September 2011, pp 1477–1484

Brizard A (2008) An introduction to lagrangian mechanics. World Scientific Publishing Company, Singapore

Cao Y, Yu W, Ren W, Chen G (2013) An overview of recent progress in the study of distributed multi-agent coordination. IEEE Trans Ind Inf 9(1):427–438

Caruntu C, Copot C, Lazar C, De Keyser R (2014) Longitudinal control of vehicle platoons for stop-and-go waves mitigation. In: Proceedings of 18th international conference on system theory, control and computing, Sinaia, Romania , 17-19 October 2014, pp 670–675

Chevalier A, Copot C, Cristescu S, Ionescu C, De Keyser R (2013a) Emulation of a highway bottleneck using leader-follower formation control. In: Proceedings of IEEE 8th international symposium on applied computational intelligence and informatics, Timisoara, Romania, 23–25 May 2013, pp 131–136

Chevalier A, Copot C, Ionescu C, De Keyser R (2013b) Emulation of highway situations using a leader-follower system. Sci Bull Politeh Univ Timis Rom Trans Autom Control Comput Sci 58(2–4):93–100

De Keyser R (2003) Model based predictive control for linear systems. In: UNESCO encyclopaedia of life support systems http://www.eolss.net, Article contribution 643161 (available online at: http://www.eolss.net/sample-chapters/c18/e6-43-16-01pdf) Eolss Publishers Co Ltd, Oxford, 35 p

De Keyser R, Ionescu C (2006) FRtool: A frequency response tool for CACSD in matlab. In: Proceedings of IEEE international symposium on computer aided control systems design, Munich, Germany, 4–6 October 2006, pp 2275–2280

Guanghua W, Deyi L, Wenyan G, Peng J (2013) Study on formation control of multi-robot systems. In: Proceedings of the 3rd international conference on intelligent system design and engineering applications, Hong Kong, 16–18 January 2013, pp 1335–1339

Hernandez A, Copot C, Cerquera J, Murcia H, De Keyser R (2014a) Formation control of UGVs using an UAV as remote vision sensor. In: Proceedings of the 19th world congress : the international federation of automatic control, Cape Town, 24–29 August 2014, pp 11,872–11,877

Hernandez A, Murcia H, Copot C, De Keyser R (2014b) Model predictive path-following control of an A.R. drone quadrotor. In: Proceedings of Memorias del XVI Congreso Latinoamericano de Control Automatico, Cancun, 14-17 October 2014, pp 618 – 623

Holmes G (2014) Introduction to the surveyor SRV-1 blackfin robot. http://gregscomputing.com/2009/10/05/introduction-to-the-surveyor-srv-1-blackfin-robot/ Cited 8 December 2014

Ionescu C, De Keyser R (2012) The next generation of relay-based pid autotuners (part 1): Some insights on the performance of simple relay-based pid autotuners. In: Proceedings of IFAC conference on advances in PID control, Brescia (Italy), 28–30 March 2012, pp 122–127
James H, Nichols N, Phillips R (1965) Theory of servomechanisms. Dover Publications, Dover
Ljung L (2007) System identification: theory for the user. Prentice-Hall, New Jersey
Maciejowski JM (2002) Predictive control: with constraints. Prentice-Hall, New Jersey
Páll E, Máthé K, Tamas L, Busoniu L (2014) Railway track following with the AR. Drone using vanishing point detection. In: IEEE international conference on automation, quality and testing, robotics, Cluj-Napoca, Romania, 22–24 May 2014, pp 1–6
Parrot (2014) ARDrone 2.0. http://www.parrot.com/nl/gallery/ardrone2/ Cited 8 December 2014
Soni Y, Bhatt R (2011) Stability and stabilization of systems with time delay. IEEE Control Syst Mag 31(1):38–65
Vlas T, Hernandez A, Copot C, Nascu I, De Keyser R (2013) Identification and path following control of an AR. Drone quadrotor. In: Proceedings of 17th international conference on system theory, control and computing (ICSTCC), Sinaia, Romania, 11–13 October 2013, pp 583–588