Robotics and Autonomous Systems 60 (2012) 657–667

Contents lists available at SciVerse ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Quadrocopter control using an on-board video system with off-board processing

Matevž Bošnak *, Drago Matko, Sašo Blažič

Faculty of Electrical Engineering, University of Ljubljana, Tržaška cesta 25, 1000 Ljubljana, Slovenia

ARTICLE INFO

Article history:
Received 20 July 2011
Received in revised form
12 October 2011
Accepted 21 October 2011
Available online 11 January 2012

Keywords:
Quadrocopter
Unmanned aerial vehicle
Position estimation
Computer vision
LQR control
Kalman filtering

ABSTRACT

In recent years, Unmanned Aerial Vehicles (UAVs) have gained increasing popularity. These vehicles are employed in many applications, from military operations to civilian tasks. One of the main fields of UAV research is the vehicle positioning problem. Fully autonomous vehicles are required to be as self-sustained as possible in terms of external sensors. To achieve this in situations where the global positioning system (GPS) does not function, computer vision can be used. This paper presents an implementation of computer vision to hold a quadrotor aircraft in a stable hovering position using a low-cost, consumer-grade, video system. The successful implementation of this system required the development of a data-fusion algorithm that uses both inertial sensors and visual system measurements for the purpose of positioning. The system design is unique in its ability to successfully handle missing and considerably delayed video system data. Finally, a control algorithm was implemented and the whole system was tested experimentally. The results suggest the successful continuation of research in this field.

1. Introduction

A quadrotor is a four-rotor helicopter. The idea of using four rotors is not new, as a full-scale, four-rotor helicopter was built by De Bothezat as early as 1921. However, quadrocopters are dynamically unstable and therefore suitable control methods are required to make them stable [1]. These control approaches normally use two control loops—a high-speed, inner loop that controls the helicopter's attitude based on the outputs from the IMU (Inertial Measurement Unit) in a *strap-down* configuration [2] and a slower, outer loop that controls the helicopter's position. While the attitude can be easily determined by measuring the acceleration due to the Earth's gravitational field and the rotational velocities, there are no universal positioning systems available. Outdoors, the GPS (Global Positioning System) that relies on receiving its signal from satellites can be used.

However, indoors, controlled objects must rely either on beacon-based systems that use RF (Radio Frequency) waves (or a combination of RF and Ultrasonic waves [3]), a combination of sensors and SLAM (simultaneous localization and mapping) algorithms [4–6] or a visual system that usually incorporates one or more video cameras. Two different video-camera arrangements are possible—a video camera fixed to an object [7–12]; a single video camera fixed in the object's environment [13] or a constellation of multiple video cameras, such as Vicon visual tracking system, which is often used in the research on the quadrocopter control [14–17].

However, real autonomous systems are required to be independent of those external sensors, but are limited by the weight of the sensors and the processing circuitry. In [18], the authors fixed a mobile phone to a quadrocopter and used its processor for localization during the flight, but there was no closed-loop control. In 2010, the company Parrot presented the AR.Drone [19], a self-stabilizing quadrocopter, controlled remotely via a phone or a computer. It combines a fast, down-facing camera, an IMU unit, an ultrasonic distance sensor and a fast signal processor to achieve its stable flight. The self-stabilizing nature makes it an attractive platform for experiments dealing with visual-based control [20]. A similar approach uses wireless transmitters to transmit the video data to an off-board computer, where image-analysis is executed and the results are passed back to the vehicle [21–24]. Our approach uses the latter configuration, but a great deal of care is taken with the inherent delays and the outages of the video-camera data with such a system.

To solve the issues with the delay and lost data, sensor-fusion methods in combination with a Kalman-filtering technique can be used [25]. In [26] different methods to solve the problem of delayed measurements are described. One such method involves extrapolating delayed measurements to the present time and still obtaining the optimality of the filter, while the other suggests updating the covariance matrix and the state at a different time. Similarly, our method uses past information about the system states, such that the states are directly compared to the measurements, but the Kalman innovation is then used in the present time to correct the states. The system uses the on-board, 6-axis, IMU unit and the image-based position and velocity data received via the wireless link. This unit measures the rotational velocities

* Corresponding author. Tel.: +386 1 47 68 702; fax: +386 1 42 64 631.
E-mail address: matevz.bosnak@fe.uni-lj.si (M. Bošnak).

0921-8890/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2011.10.009

Fig. 1. The X-3D-BL quadrocopter.

around the principal axes, x, y and z, and the accelerations in the inertial reference frame. The accelerations are translated into the base reference frame (fixed to the target on the ground) and used for the prediction stage of the Kalman filter. In order to boost the performance of the filter implementation on a small microcontroller, a linear steady-state variant of Kalman filter was designed with minimal number of operations over matrices. Additionally, the Kalman filter correction stage with the delay-cancelling technique is executed only when the new image-based position data is received, while the prediction stage of the Kalman filter is calculated every 10 ms on the on-board computer, making this system resilient to short (in the range of a few seconds) communication interruptions.

This paper presents a slightly modified approach to visual servo-control that provides promising results. The initial analysis of the quadrocopter's dynamics is provided; this serves as a basis for the Kalman-filtering technique described later on. The innovative system for delay approximation is then presented, which complements the initial Kalman filter. Finally, the visual servo-control was evaluated during the stationary flight of the quadrocopter and the results of the experiment are presented.

2. The description of the quadrotor system

The proposed approach was implemented on a quadrotor helicopter X-3D-BL (quadrocopter), originally presented in [14]. The X-3D-BL is a commercial product and serves as a testbed for many experiments ([15,7,18,27,16] and others). The quadrocopter (shown in Fig. 1) has a classical four rotor design with two counter rotating pairs of propellers arranged in a square and connected to the cross of the diagonals. The controller board, including sensors, is mounted in the middle of the cross together with the battery, while the brushless controllers are mounted on the underside of the booms. As the quadrocopter is operated indoors, a lightweight shield is mounted around the propellers for the additional safety. For the purpose of the research, the quadrocopter was equipped with the KX171 color video camera, video wireless transmitter, camera power supply and the timestamp generator circuit (overview of the complete system is given in Fig. 2). The total take-off weight of the quadrocopter is 680 g.

Any treatment of dynamic quantities involves the use of coordinate systems. Let us define two coordinate systems that are directly related to the experiment in this paper. The target coordinate system $\mathcal{T}$ (illustrated in Fig. 3) is a standard, right-handed, Cartesian coordinate system that has its origin at the target's center, the principal x and y axes in the plane of the target and the axes z in the vertical up direction. The target is visually asymmetrical and therefore the target coordinate system can be uniquely defined. The quadrocopter's coordinate system $\mathcal{K}$ (also illustrated in Fig. 3) has its origin at the center of the quadrocopter's frame, with the axes aligned with the quadrocopter rotors' booms. The axis x is oriented towards the front boom of the quadrocopter, the axis y towards the left boom and the z axis faces upwards along the antenna so that $\mathcal{K}$ also defines a right-handed Cartesian coordinate system. This orientation was selected because of the acceleration sensors' orientation.

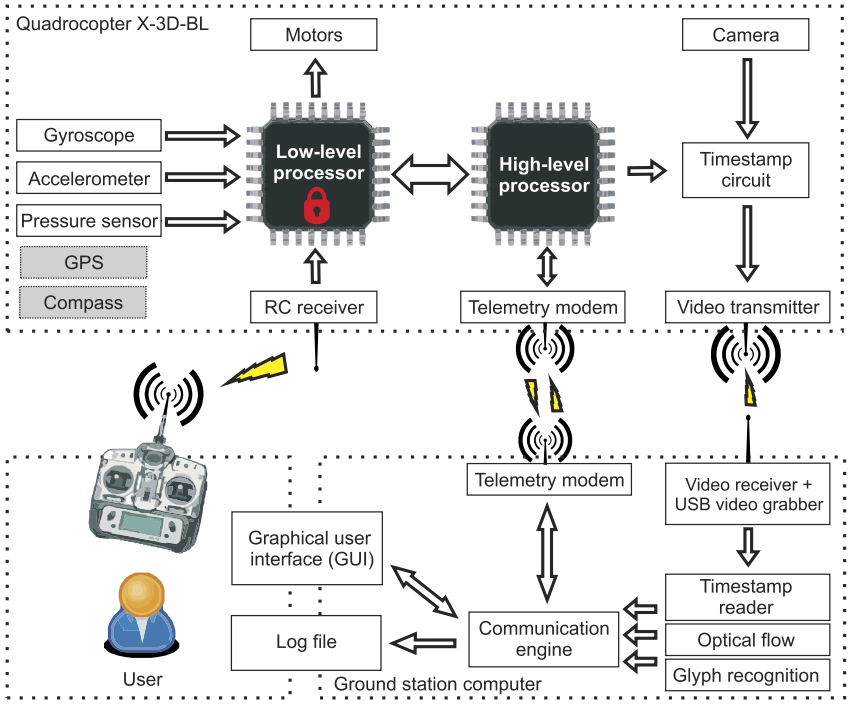


Fig. 2. Illustration of the complete system.

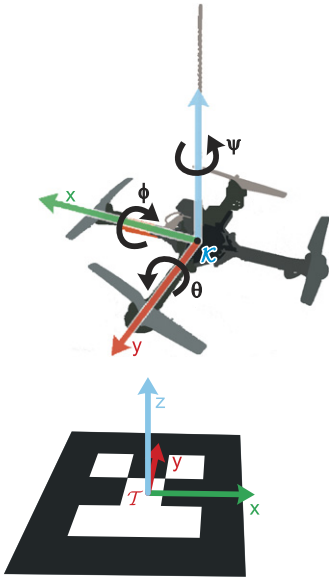


Fig. 3. Graphical representation of the coordinate systems used.

Let $\mathbf{R}$ be the rotational transformation between $\mathcal{K}$ and $\mathcal{T}$, so that $\mathbf{R}: \mathcal{K} \rightarrow \mathcal{T}$. As the quadrocopter's navigation is based only on the target coordinate system, the global coordinate system will not be dealt with in this paper.

The X-3D-BL quadrocopter used in the experiment was already equipped with a low-level stabilization loop hosted in the low-level microprocessor that controls the quadrocopter's attitude. It uses four, in pairs, counter-rotating brushless motors with the appropriate brushless-motor controllers. By delivering power to each motor independently, the resulting thrust can be modulated as in the case of a "normal" helicopter with swashplate-controlled rotors. The overall thrust is controlled by the combined speed of all four rotors, and the rotation around the vertical z-axis (the yaw angle ψ) by increasing one pair of rotors and decreasing the other (the counter-rotating) to the first. The pitch θ and roll ϕ angles are controlled similarly by modulating the power to the front and back rotors (for pitch) or the left and right rotors (for roll).

The high-level microprocessor has access to the attitude information of the low-level processor (measured values of the angles pitch θ, roll ϕ and yaw ψ), the calibrated measurements of the acceleration, the gyroscope and the magnetic sensors and can take over the control of the attitude and thrust commands.

The equations of motion for the quadrocopter can be derived separately for translational and rotational motion. The rotational motion part of these equations is already implemented in the low-level processor and it keeps track of the quadrocopter's attitude. Since our goal is to estimate and control the position of the quadrocopter, the rotation dynamics will not be treated.

Let $\vec{p}_{\mathcal{T}} = (x, y, z)$ be the position of the center of mass of the quadrocopter in the target coordinate system. Then, Newton's equations of translational motion from the external point of view can be written as

$$\frac{d^2}{dt^2}\vec{p}_{\mathcal{T}} = \frac{1}{m}\left(\vec{F}_{\text{th},\mathcal{K}}\mathbf{R}(\phi, \theta, \psi) - m\vec{g} - b\vec{v}_{\mathcal{T}}\right) \tag{1}$$

where $\vec{F}_{\text{th},\mathcal{K}}$ is the total thrust vector in the quadrocopter coordinate system, m is the mass of the quadrocopter treated as a solid body, $\vec{g}$ is the gravitational acceleration vector, $\vec{v}_{\mathcal{T}}$ is the speed of the quadrocopter in the target coordinate system, b is a damping factor due to air resistance at slow speeds (linear drag is assumed) and

$$\mathbf{R}(\phi, \theta, \psi) = \mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\psi) \tag{2}$$

where $\mathbf{R}_x(\phi)$ is a rotational matrix about the x-axis, $\mathbf{R}_y(\theta)$ is a rotational matrix about the y-axis and $\mathbf{R}_z(\psi)$ is a rotational matrix about the z-axis of the quadrocopter coordinate system $\mathcal{K}$. The onboard acceleration sensor measures both the static (due to gravitation) and dynamic accelerations in the quadrocopter coordinate system $\mathcal{K}$. Therefore, to estimate the position of the quadrocopter, based only on the dynamic component of the acceleration measurement, the following must be evaluated

$$\frac{d^2}{dt^2}\vec{p}_{\mathcal{T}} = \vec{a}_{\mathcal{K}}\mathbf{R} - \vec{g} \tag{3}$$

where $\vec{a}_{\mathcal{K}}$ is the acceleration vector in the quadrocopter's coordinate system $\mathcal{K}$ and $\vec{g}$ is the gravitational acceleration vector in the target coordinate system $\mathcal{T}$. Although the acceleration sensor is calibrated, its output is drifting during normal operation due to various effects (including temperature changes and supply-voltage fluctuations), which cannot be totally eliminated. Therefore, it is necessary to take the sensor's output bias into account. The acceleration biases $\vec{a}_b$ (given in $\mathcal{K}$) must be estimated and then subtracted from the sensor reading $\vec{a}_s$ (also given in $\mathcal{K}$) as in

$$\frac{d^2}{dt^2}\vec{p}_{\mathcal{T}} = \left(\vec{a}_s - \vec{a}_b\right)\mathbf{R} - \vec{g}. \tag{4}$$

3. Computer-vision system

Instead of relying on external positioning systems, our study is aimed at using the video camera as the primary sensor for positioning purposes. To avoid putting a powerful and heavy computer on-board the aircraft, video is wirelessly transferred to a standard personal computer, running our custom video-recognition software. The software, developed in Visual C# using the combination of AForge.NET and Emgu CV frameworks for image processing and glyph Recognition library GRATF [28] for glyph extraction, estimates the optical flow vectors and extracts the target's position, which is used to calculate the position of the origin and the orientation of $\mathcal{K}$ relative to $\mathcal{T}$. The vision system works in three modes:

1. Mode 0: Not enough image data is available for either the optical flow analysis or the target following—no measurements are available.
2. Mode 1: Optical flow analysis is operational, but the target glyph is not in video camera view—only speed measurements are available.
3. Mode 2: Optical flow analysis and target following are fully operational—position and speed measurements are available.

3.1. Optical flow analysis

By attaching a camera to a moving object, a set of fixed points in space, observed by the camera, is projected into a set of 2D points $\vec{\mathbf{x}}_i^{\text{OF}}(t) = (x_i^{\text{OF}}(t), y_i^{\text{OF}}(t))$ moving along their 2D paths, the instantaneous derivative of which is the velocity $d\vec{\mathbf{x}}_i^{\text{OF}}(t)/dt$. The field of 2D velocities for all the visible surface points is often referred to as the 2D motion field [29]. The goal of the optical flow estimation is to compute an approximation of the motion field from the time-varying position of the image-features.

In our experiment, a wireless camera is used that outputs an interlaced PAL video signal. This interlaced video signal is a legacy of the analogue TV sets, where each frame consists of odd and even fields, one for odd and one for even lines, shot at different times, which are displayed on the analogue screen at a double frame rate. Because the fields are not captured at the same time, any motion in the image (resulting from the motion of the object in view or motion of the camera) results in the production of

Fig. 4. Interlaced image of the target during the camera movement.

jagged edges in the image features (demonstrated in Fig. 4). The interlacing effects usually cause a lot of problems in video-signal processing and various techniques are employed to remove the artifacts in the process of de-interlacing before the frame is fed to the image-analysis system. However, we can exploit this fact to estimate the motion field vectors by analyzing the apparent motion of the same features between two fields. By measuring the angular velocity of the camera and the distance to the object by other means, the feature velocity with respect to the camera can be accurately estimated.

Each grabbed video frame was first separated into odd and even fields. Then FAST corner detection [30] with variable threshold was executed on the odd field and the resulting features were searched in the even field using the iterative Lucas–Kanade method in pyramids [31] to calculate the sparse motion field vectors. The variable threshold method was used in order to produce approximately the same number of FAST corners, which are normally affected by the complexity of the texture in the grabbed video frame. If the number of detected FAST corners was below a predefined value, the FAST corner detection threshold was reduced. Similarly, the threshold was increased if too many corners were detected.

The resulting motion field vectors were then averaged by the lengths and the orientations. All vectors that had the length or the orientation distinctly different than the average were processed as outliers and were discarded. The remaining vectors were used to recalculate the average $(\Delta x^{\mathrm{OF}}, \Delta y^{\mathrm{OF}})$, which was then used as the estimation of the optical flow due to the camera's combined lateral $(\Delta x_m^{\mathrm{OF}}, \Delta y_m^{\mathrm{OF}})$ and angular $(\Delta x_r^{\mathrm{OF}}, \Delta y_r^{\mathrm{OF}})$ motion

$$(\Delta x^{\mathrm{OF}}, \Delta y^{\mathrm{OF}}) = \lambda \frac{(\Delta x_m^{\mathrm{OF}}, \Delta y_m^{\mathrm{OF}}) + (\Delta x_r^{\mathrm{OF}}, \Delta y_r^{\mathrm{OF}})}{z} \tag{5}$$

where λ is the constant that links the pixel coordinates with the real-world coordinates and z is the distance from the camera's origin to the horizontal plane, where the features are observed.

The apparent lateral motion of the camera due to the camera's rotation $(\Delta x_r^{\mathrm{OF}}, \Delta y_r^{\mathrm{OF}})$ can then be safely approximated by the circular motion of the features at the radius z

$$\Delta x_r^{\mathrm{OF}} = z\omega_\phi \Delta t \tag{6}$$

$$\Delta y_r^{\mathrm{OF}} = z\omega_\theta \Delta t \tag{7}$$

where ω_θ and ω_ϕ are the radial velocities about the x and y-axes of the $\mathcal{K}$ coordinate system and Δt is the time interval between the odd and the even field of the video frame.

By inserting (6), (7) into (5) and some rearranging, the lateral motion of the camera can be expressed as

$$\Delta x_m^{\mathrm{OF}} = zK_1^{\mathrm{OF}} \left[\Delta x^{\mathrm{OF}} - K_2^{\mathrm{OF}} \omega_\phi\right] \tag{8}$$

$$\Delta y_m^{\mathrm{OF}} = zK_1^{\mathrm{OF}} \left[\Delta y^{\mathrm{OF}} - K_2^{\mathrm{OF}} \omega_\theta\right] \tag{9}$$

where K_1^{OF} and K_2^{OF} are constants that are obtained with the camera's calibration. Eqs. (8) and (9) show that the angular motion of the camera can be directly subtracted from the observed total motion without the need to know the distance between the camera and the observed features. Another advantage of this approach is that the angular motion of the camera is measured directly by the gyroscope sensor. However, for an exact absolute lateral motion of the camera, the distance z must be measured by other means.

3.2. Target recognition and position determination

The target coordinate system is marked with a target graphical glyph, which is represented by a black border and a square central grid, divided equally into an equal number of rows and columns. The central glyph cells are coded in white or black for the identification of each glyph. The target glyph code is entered into the glyph database and when a match between a glyph in the current video-camera frame and an entry in the database is found, the glyph's position and size are used in the following position-determination procedure.

Instead of determining the homography matrix between the target glyph plane and the camera plane, we followed a more specialized approach, taking advantage of the sensory data already present in the system. In the first step we assume that the camera frame is leveled horizontally, with the camera's central axis directed straight down and the camera's viewpoint being at the origin of the $\mathcal{K}$ coordinate system. Also, we assume that the projective geometry of the pinhole camera is modeled by a perspective projection [32] with an additional radial distortion due to the wide-angle lens mounted on the camera.

Wide-angle camera lenses are commonly used in the field of computer vision; therefore, the problem of camera calibration has received much attention in computer-vision applications. The most frequently used method is the polynomial model for camera-distortion removal, but [33] suggested a different mathematical model for radial distortion based on a camera-and-lens projection geometry. Their idea is presented and followed in the approach below.

To correct the radial distortion, a corresponding model based on the camera-and-lens projection geometry is used:

$$R = f(r) = \frac{H}{2} \frac{1 - e^{-2r/H}}{e^{-r/H}} = H \sinh \frac{r}{H} \tag{10}$$

where R is the rectified radius, r is the radius from the distorted image, defined in (11), and H is the parameter of the lens, which is determined with the camera calibration.

Let C_x and C_y be the coordinates of a pixel in the distorted image, and let C_x' and C_y' be the coordinates of the same pixel in the rectified image. The origin of the transformation (10) is placed in the center of the image. The relations between (C_x, C_y) and (C_x', C_y') are as follows:

$$\begin{aligned} r &= \sqrt{(C_x - C_x^p)^2 + (C_y - C_y^p)^2}, \\ \varphi &= \arctan 2 \frac{C_y - C_y^p}{C_x - C_x^p}, \\ C_x' &= R \cos \varphi, \\ C_y' &= R \sin \varphi \end{aligned} \tag{11}$$

where (C_x^p, C_y^p) are the coordinates of the pixel in the center of the image and arctan 2 is the four-quadrant version of the inverse tangent function.

After the camera-distortion rectification is performed, a point on the target plane (T_x, T_y, T_z) (in Fig. 5 this point's projection onto the x–z plane is illustrated), whose coordinates are expressed with

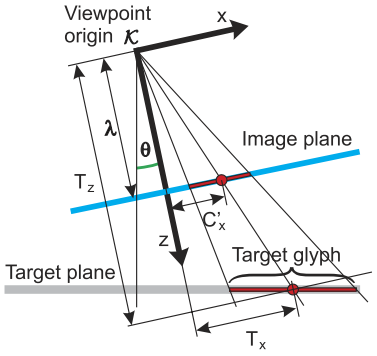


Fig. 5. Calculating the position of the origin of the $\mathcal{K}$ in the $\mathcal{T}$ coordinate system.

respect to the $\mathcal{K}$ coordinate system, will project onto the image plane at a point C'_x, C'_y, given by

$$\begin{bmatrix} C'_x \\ C'_y \end{bmatrix} = \frac{\lambda \nu}{T_z} \begin{bmatrix} T_x \\ T_y \end{bmatrix} \tag{12}$$

where λ denotes the distance of the viewpoint origin behind the image plane [10], ν is the pixel density (in pixels per millimeter) and T_z is the distance between the viewpoint origin and the target plane, perpendicular to the image plane. The value of the variable T_z is calculated from the size of the recognized glyph $(\Delta T_x, \Delta T_y)$, which is compared to the size of the glyph in the image $(\Delta C_x, \Delta C_y)$.

$$T_z = \lambda \nu \sqrt{\frac{\Delta T_x^2 + \Delta T_y^2}{\Delta C_x^2 + \Delta C_y^2}} = \lambda \nu \frac{a_T}{a_C} \tag{13}$$

where a_T is the size of the target in millimeters and a_C is the size of the image of the target in pixels. By combining (12) and (13), the coordinates T_x and T_y can then be obtained

$$\begin{bmatrix} T_x \\ T_y \end{bmatrix} = \frac{a_T}{a_C} \begin{bmatrix} C'_x \\ C'_y \end{bmatrix}. \tag{14}$$

The image of the glyph also defines the z-orientation of the quadrocopter's coordinate system with regard to the target coordinate system. The angle ψ is calculated from the angle of the glyph's diagonal, connecting the top-right $(T_{x,0}, T_{y,0})$ and the bottom-left $(T_{x,2}, T_{y,2})$ corners of the glyph.

$$\psi = \arctan 2 \frac{T_{y,0} - T_{y,2}}{T_{x,0} - T_{x,2}}. \tag{15}$$

To determine the true position of $\mathcal{K}$ with respect to $\mathcal{T}$ the effect of the video-camera tilt and orientation is compensated on-board the quadrocopter. As the target position data is delayed for d due to image transmission, processing and communication delays, the pitch, roll and yaw angles are delayed by the same amount. Then the point $\mathbf{T}$ is rotated about the x-axis by the angle $\phi(k-d)$, about the y-axis by the angle $\theta(k-d)$ and about the z-axis by the angle $\psi(k-d)$.

$$\begin{aligned}(T_x^f(k), T_y^f(k), T_z^f(k)) &= \mathbf{R}_z(\psi(k-d))\mathbf{R}_y(\theta(k-d)) \\ &\quad \times \mathbf{R}_x(\psi(k-d)) \begin{bmatrix} T_x(k) \\ T_y(k) \\ T_z(k) \end{bmatrix}\end{aligned} \tag{16}$$

where $\mathbf{R}_x(\psi)$, $\mathbf{R}_y(\theta)$ and $\mathbf{R}_z(\psi)$ are the standard rotation matrices about the axes of rotation, x, y and z, respectively. The coordinates $(-T_x^f, -T_y^f, -T_z^f)$ finally define the origin of $\mathcal{K}$ with respect to $\mathcal{T}$.

3.3. Video-system delay estimation

To deal with the variable delay of the video system, the quadrocopter was equipped with an additional circuit that embeds the binary-coded value of the on-board 8-bit timer into every video frame. The timestamp overlay is produced using a small PIC microprocessor that encodes each bit of the current timer value as a series of white and black stripes. This encoding scheme was used for its decoding simplicity and robustness. The timestamp data is extracted in the computer software and transmitted together with other image-processing results wirelessly back to the quadrocopter microprocessor, where the data is compared with the on-board timer and the exact delay is calculated. This delay estimate comprises of the vision-system transmission and the processing delay, together with the communication delay. The resolution of the encoded timer value is 10 ms, so the system is able to identify the delays in the range from 0 to approximately 2.5 s.

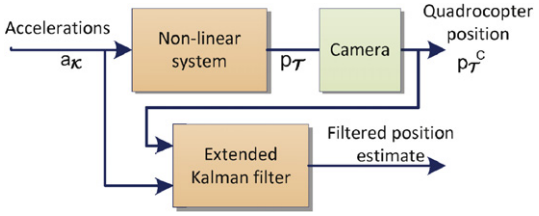


Fig. 6. Block diagram of the system with the extended Kalman filter.

4. Position estimation

Position information, produced by the image recognition, is subjected to delays and signal outages before it reaches the control input of the quadrocopter. One major drawback of direct visual servoing is the need for the target to stay inside the field of view of the camera. In the case of the quadrocopter under direct visual servo control, one must be aware of the unstable nature of this aircraft. This nature requires a functioning visual servoing loop, which is directly dependent on the visibility of the target by the camera. Therefore, an indirect visual servo-control was developed that uses a combination of local position tracking with an integrated IMU unit and an image-based position estimation and filtering with a Kalman filter. The additional dynamic delay estimation and compensation system was included in the position filtering process.

As the quadrocopter, as a system, includes nonlinearities, it is common practice to employ the Extended Kalman filter (illustrated in Fig. 6), where a linear approximation is only used for solving the Riccati equation, a result of which is the Kalman gain. The full, nonlinear model is used in the propagation of the estimate and in computing the predicted sensor outputs [34]. This would introduce a heavy load on the on-board, high-level processor and thus was not selected for our application. Therefore, a nonlinear part of the quadrocopter system (mainly the nonlinear coordinate system transformation from the quadrocopter's coordinate system $\mathcal{K}$ to the target coordinate system $\mathcal{T}$) was decoupled from the linear part of the system and replicated on the path of the acceleration measurement vector $\mathbf{a}_\mathcal{K}$ entering the Kalman filter (Fig. 7) in order to enable $\mathbf{a}_\mathcal{T}$ to enter the Kalman filter directly. This enabled us to employ a basic (linear) form of the Kalman filter that presents a much lighter load to the processor.

4.1. Position prediction

The position prediction is accomplished by measuring and integrating the dynamic acceleration of the quadrocopter. Unfortunately, the inertial sensors measure the total acceleration, combining the effect of the dynamic acceleration due to the velocity change and the effect of gravitational acceleration. To successfully isolate the dynamic component from the sensor readings, the sensor biases are first subtracted from the measured

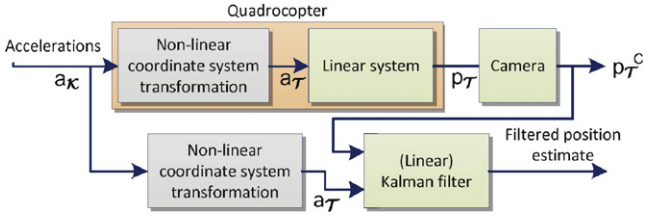


Fig. 7. Block diagram of the system with the linear Kalman filter.

acceleration in the $\mathcal{K}$ coordinate system. Then, the resulting acceleration vector is transformed to the target coordinate system $\mathcal{T}$ (in our experiments, the target coordinate system is, due to target's fixed position, effectively the world coordinate system), where the effect of gravity is known and can be subtracted from the readings. The transformation of the acceleration has one other major advantage—position prediction produces the position of the quadrocopter directly in the target coordinate system $\mathcal{T}$ and the use of nonlinearities is avoided in further processing, and the Kalman filter, used to correct the position of the quadrocopter, can be simplified.

In our experiment, the position prediction is all done on-board the quadrocopter in the high-level processor. The calibrated acceleration sensor readings $\mathbf{a}_{s,\mathcal{K}}(k)$ and the current Euler angles ϕ, θ and ψ are transferred periodically from the low-level processor to the high-level processor, where the DCM rotation matrix is constructed. As the position of the x and y axes is swapped, the DCM rotation matrix has a slightly modified form

$$\mathbf{R}_{\mathrm{DCM}} = \begin{bmatrix} c\phi s\psi + c\psi s\phi s\theta & c\phi c\psi - s\psi s\phi s\theta & -c\theta s\phi \\ c\theta c\psi & -c\theta s\psi & s\theta \\ s\phi s\psi - c\phi s\theta c\psi & s\phi c\psi + c\phi s\theta s\psi & c\phi c\theta \end{bmatrix} \tag{17}$$

where these abbreviations were used

$$\begin{aligned} c\phi &= \cos\phi \quad & c\theta &= \cos\theta \quad & c\psi &= \cos\psi \\ s\phi &= \sin\phi \quad & s\theta &= \sin\theta \quad & s\psi &= \sin\psi. \end{aligned} \tag{18}$$

With the above DCM matrix the dynamic acceleration vector $a_{d,\mathcal{T}}$ in the $\mathcal{T}$ coordinate system was extracted

$$\begin{aligned} \mathbf{a}_{d,\mathcal{T}}(k) &= \begin{bmatrix} a_{d,\mathcal{T},x}(k) \\ a_{d,\mathcal{T},y}(k) \\ a_{d,\mathcal{T},z}(k) \end{bmatrix} \\ &= \left(\left(\mathbf{a}_{s,\mathcal{K}}(k) - \mathbf{b}_{\mathcal{K}}(k) \right)^T \mathbf{R}_{\mathrm{DCM}}(k) - \begin{bmatrix} 0 & 0 & g \end{bmatrix} \right)^T \end{aligned} \tag{19}$$

where the calibrated acceleration-sensor-readings vector $\mathbf{a}_{s,\mathcal{K}}(k)$ and the acceleration-sensor-bias vector $\mathbf{b}_{\mathcal{K}}(k)$ are defined as

$$\mathbf{a}_{s,\mathcal{K}}(k) = \begin{bmatrix} a_{s,\mathcal{K},x} \\ a_{s,\mathcal{K},y} \\ a_{s,\mathcal{K},z} \end{bmatrix} \qquad \mathbf{b}_{\mathcal{K}}(k) = \begin{bmatrix} b_{s,\mathcal{K},x} \\ b_{s,\mathcal{K},y} \\ b_{s,\mathcal{K},z} \end{bmatrix}. \tag{20}$$

The resulting dynamic acceleration is then integrated twice using the Euler method at a rate of 100 times per second. As the sensor biases $\mathbf{b}_{\mathcal{K}}(k)$ are unknown and variable, three additional states for them were added to the system state vector

$$\mathbf{x}(k) = \begin{bmatrix} p_{\mathcal{T},x}(k) \\ v_{\mathcal{T},x}(k) \\ b_{\mathcal{K},x}(k) \\ p_{\mathcal{T},y}(k) \\ v_{\mathcal{T},y}(k) \\ b_{\mathcal{K},y}(k) \\ p_{\mathcal{T},z}(k) \\ v_{\mathcal{T},z}(k) \\ b_{\mathcal{K},z}(k) \end{bmatrix} \tag{21}$$

where $p_{\mathcal{T},x}$, $p_{\mathcal{T},y}$ and $p_{\mathcal{T},z}$ are the positions in the x, y and z axes, $v_{\mathcal{T},x}$, $v_{\mathcal{T},y}$ and $v_{\mathcal{T},z}$ are the velocities along the same axes, and $b_{\mathcal{K},x}$, $b_{\mathcal{K},y}$ and $b_{\mathcal{K},z}$ are the acceleration sensor biases for all three axes in the $\mathcal{K}$ coordinate system. The system can then be written in state-space form as

$$\mathbf{x}(k+1) = \begin{bmatrix} \mathbf{A}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_0 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} \mathbf{B}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_0 \end{bmatrix} \mathbf{a}_{d,\mathcal{T}}(k) \tag{22}$$

$$\mathbf{y}(k) = \begin{bmatrix} \mathbf{C}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{C}_0 \end{bmatrix} \mathbf{x}(k) \tag{23}$$

where

$$\mathbf{A}_0 = \begin{bmatrix} 1 & T & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{B}_0 = \begin{bmatrix} \frac{T^2}{2} \\ T \\ 0 \end{bmatrix} \qquad \mathbf{C}_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{24}$$

If (19) is inserted into (22), the following expression is produced

$$\begin{aligned} \mathbf{x}(k+1) = &\left(\begin{bmatrix} \mathbf{A}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_0 \end{bmatrix} \mathbf{x}(k) \right. \\ &\left. - \begin{bmatrix} \mathbf{B}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_0 \end{bmatrix} \mathbf{R}_{\mathrm{DCM}}^T(k) \mathbf{b}_{\mathcal{K}}(k) \right) \\ &+ \begin{bmatrix} \mathbf{B}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_0 \end{bmatrix} \left(\mathbf{R}_{\mathrm{DCM}}^T(k) \mathbf{a}_{s,\mathcal{K}}(k) - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \right). \end{aligned} \tag{25}$$

This system predicts the quadrocopter's velocity and position. However, the integration causes errors to accumulate and the velocity and position predictions tend to drift.

4.2. Kalman filtering

While the visual tracking system, as mentioned before, suffers because of a low update rate and the long delay caused by the image-capture and analysis procedure, the position-prediction stage exhibits a strong tendency to drift over time. The Kalman-filtering technique was used to fuse the estimates of both systems together with the aim to take advantages of them both.

As can be seen from Eqs. (21)–(24), each of the axes can be inspected independently of each other. Here, we only show the solutions for the x- and the z-axis, because the solution for the y-axis is identical to the solution for the x-axis.

The state vector for the x-axis is defined as follows

$$\mathbf{x}_x(k) = \begin{bmatrix} p_{\mathcal{T},x}(k) \\ v_{\mathcal{T},x}(k) \\ b_{\mathcal{K},x}(k) \end{bmatrix}. \tag{26}$$

The system and measurement processes are affected by the process noise $\mathbf{w}_x$ and the measurement noise n_x, which are assumed to be independent of each other, white, and with normal probability distributions [35]. This produces the following set of equations in the system space

$$\mathbf{x}_x(k+1) = \mathbf{A_0}\mathbf{x}_x(k) + \mathbf{B_0}a_{d,\mathcal{T},x}(k) + \mathbf{F}_x\mathbf{w}_x(k) \tag{27}$$

$$y_x(k) = \mathbf{C}_0\mathbf{x}_x(k) + n_x(k) \tag{28}$$

where $\mathbf{F}_x$ is a matrix of the appropriate size.

Usually, the Kalman filter is implemented by executing the following five steps:

1. $\mathbf{x}_x^*(k+1) = \mathbf{A}_0\hat{\mathbf{x}}_x(k) + \mathbf{B}_0 a_{d,\mathcal{T},x}(k)$.
2. $\mathbf{P}_x^*(k+1) = \mathbf{A}_0\hat{\mathbf{P}}_x(k)\mathbf{A}_0^T + \mathbf{F}_x\mathbf{V}_x\mathbf{F}_x^T$.

3. In the new time step k : $\mathbf{K}_x(k) = \mathbf{P}^*_x(k)\mathbf{C}_0^T[\mathbf{C}_0\mathbf{P}^*_x(k)\mathbf{C}_0^T + \mathbf{N}_x]^{-1}$.
4. $\hat{\mathbf{x}}_x(k) = \mathbf{x}^*_x(k) + \mathbf{K}_x(k)[y_x(k) - \mathbf{C}_0\mathbf{x}^*_x(k)]$.
5. $\hat{\mathbf{P}}_x(k) = \mathbf{P}^*_x(k) - \mathbf{K}_x(k)\mathbf{C}_0\mathbf{P}^*_x(k)$

where the following symbols are used:

$\mathbf{x}^*_x$—state-vector prediction,
$\hat{\mathbf{x}}_x$—state-vector estimation,
$\mathbf{P}^*_x$—covariance-matrix prediction,
$\hat{\mathbf{P}}_x$—covariance-matrix estimation,
$\mathbf{V}_x$—covariance matrix of the process noise $\mathbf{w}_x$,
$\mathbf{F}_x$—input matrix of the process noise,
$\mathbf{N}_x$—covariance matrix of the measurement noise $\mathbf{n}_x(k)$.

For the purpose of implementing linear Kalman filtering, the system had to be augmented to take into account the nonlinear properties of the sensor-bias states. The expression (25) is linearized in the nominal operating point (the target and quadrocopter frames are parallel) and the $\mathbf{R}_{\mathrm{DCM}}$ is assumed to be a unitary matrix. Due to the fact that $\mathbf{b}_{\mathcal{K},x}(k)$ is the last element of the state vector $\mathbf{x}_x(k)$, the following can be concluded from (25):

$$\left.\frac{\partial \mathbf{x}_x(k+1)}{\partial \mathbf{x}_x(k)}\right|_{\text{nom. op. point}} = \mathbf{A}_0 - \mathbf{B}_0\begin{bmatrix}0 & 0 & 1\end{bmatrix} = \mathbf{A}_{L,0} = \begin{bmatrix}1 & T & -\frac{T^2}{2}\\ 0 & 1 & -T\\ 0 & 0 & 1\end{bmatrix} \tag{29}$$

allowing the bias state to become observable in order to produce the appropriate Kalman gains.

By assuming that the system defined by (27), (28) is not time-varying and the covariance matrices $\mathbf{N}_x$ and $\mathbf{V}_x$ are constant, the steps 2, 3 and 5 of the above algorithm need not be evaluated on-line during each calculation step. Rather, the solution of the steady-state matrix Riccati equation can be found:

1. $\overline{\mathbf{P}^*}_x = \mathbf{A}_{L,0}\bar{\hat{\mathbf{P}}}_x\mathbf{A}_{L,0}^T + \mathbf{F}_x\mathbf{V}_x\mathbf{F}_x^T$.
2. $\overline{\mathbf{K}}_x = \overline{\mathbf{P}^*}_x\mathbf{C}_0^T[\mathbf{C}_0\overline{\mathbf{P}^*}_x\mathbf{C}_0^T + \mathbf{N}_x]^{-1}$.
3. $\bar{\hat{\mathbf{P}}}_x = \overline{\mathbf{P}^*}_x - \overline{\mathbf{K}}_x\mathbf{C}_0\overline{\mathbf{P}^*}_x$.

The estimation algorithm then becomes much simpler:

1. $\mathbf{x}^*_x(k+1) = \mathbf{A}_0\hat{\mathbf{x}}_x(k) + \mathbf{B}_0 a_{d,\mathcal{T},x}(k)$
2. $\hat{\mathbf{x}}_x(k) = \mathbf{x}^*_x(k) + \overline{\mathbf{K}}_x[y_x(k) - \mathbf{C}_0\mathbf{x}^*_x(k)]$.

In all the filter variations below, the following covariance matrix of the process noise is used:

$$\mathbf{V}_0 = \begin{bmatrix}\sigma_p^2 & 0 & 0\\ 0 & \sigma_v^2 & 0\\ 0 & 0 & \sigma_b^2\end{bmatrix} \tag{30}$$

where $\sigma_p^2 = 0.1\ \mathrm{cm}^2$, $\sigma_v^2 = 0.1\ \mathrm{cm}^2/\mathrm{s}^2$ and $\sigma_b^2 = 0.001\ \mathrm{cm}^2/\mathrm{s}^4$. The variances σ_p^2 and σ_v^2 reflect the system model uncertainties due to Euler integration method used, while σ_b^2 is used to model the sensor bias with a random-walk model [36]. Finally, all three noise constants were finely tuned by means of experiments in order to achieve the best operation of the filtering system.

For each of the vision-system operating modes (see Section 3), the structure of the filter is adjusted as follows:

1. Mode 0: No measurement is available for the x- and y-axis. The z-axis velocity measurement is available by measuring the air-pressure changes. With the estimation of the measurement noise at $N_{\dot{z}} = 100\ \mathrm{cm}^2/\mathrm{s}^2$ and $\mathbf{C}_0 = \begin{bmatrix}0 & 1 & 0\end{bmatrix}$, the following Kalman-filter gain was calculated

$$\overline{\mathbf{K}}_{\dot{z}} = \begin{bmatrix}0.0098 & 0.0321 & 0.0031\end{bmatrix}^T. \tag{31}$$

2. Mode 1: Velocity measurements are available for all three axes. Measurements for the x- and y-axis are based on the tilt-motion compensated, optical flow, velocity results, while the measurement for the z-axis is based on air-pressure change sensor readings. For all three axes the matrix $\mathbf{C}_0$ has the value $\mathbf{C}_0 = \begin{bmatrix}0 & 1 & 0\end{bmatrix}$. The measurement noise for the z-axis and the corresponding Kalman-filter parameters are the same as in operation mode 0, while the measurement noise for the velocity in the x- and y-axis was estimated at $N_{\dot{x},\dot{y}} = 10\ \mathrm{cm}^2/\mathrm{s}^2$ (the values were estimated based on the results of an experiment where the quadrocopter was hovering at $z = 70$ cm). The Kalman-filter gain for the x- and y-axis therefore has the value

$$\overline{\mathbf{K}}_{\dot{x},\dot{y}} = \begin{bmatrix}0.0095 & 0.0960 & 0.0095\end{bmatrix}^T. \tag{32}$$

3. Mode 2: Additional camera-position measurements are available. Because the measurements of the air-pressure changes are available during every correction step and have no delay, while the vision-system measurements are delayed and less frequent, the z-axis measurements of velocity and position are handled separately using the sequential sensor model [25]. The Kalman filter for the estimation of $\dot{z}$ is identical to the one presented in mode 0. The other two velocities and all three positions are measured by the camera. In the case of the x and the y axes, the positions and the velocities are measured and the corresponding covariance matrix of the measurement noise and the resulting Kalman gain, respectively, are

$$\mathbf{N}_{x,y} = \begin{bmatrix}2\ \mathrm{cm}^2 & 0\\ 0 & 10\ \mathrm{cm}^2/\mathrm{s}^2\end{bmatrix}$$
$$\overline{\mathbf{K}}_{x,y} = \begin{bmatrix}0.2004 & 0.0126 & 0.0013\\ 0.0025 & 0.0958 & 0.0095\end{bmatrix}^T. \tag{33}$$

In the case of the z-axis, only the position is measured by the camera, where the corresponding covariance matrix of the measurement noise and the resulting Kalman gain are

$$\mathbf{N}_z = \begin{bmatrix}2\ \mathrm{cm}^2\end{bmatrix} \qquad \overline{\mathbf{K}}_z = \begin{bmatrix}0.2085 & 0.2188 & 0.0199\end{bmatrix}^T. \tag{34}$$

The details on compensating the Kalman filter parameters for the delay will be explained in Section 4.3.

Due to the assumption of $\mathbf{R}_{\mathrm{DCM}}$ being a unitary matrix and incorporating the effect of sensor acceleration biases into the matrix $\mathbf{A}_{L,0}$ (29), an additional step must be included to correctly update the bias state in the state vector. This was accomplished by transforming the bias innovations in all three axes with $\mathbf{R}_{\mathrm{DCM}}$ from $\mathcal{T}$ back to $\mathcal{K}$ before the state vector was updated.

Such an implementation of the Kalman filter had a significantly lower complexity of implementation on the on-board, high-level processor as the number of operations over the matrices was greatly reduced and no matrix had to be inverted. The state prediction, as the first step in the filter, was already being made by the position-tracking system described in 4.1 and thus the Kalman filtering was reduced solely to the correction step.

4.3. Dynamic-delay estimation and compensation

The visual recognition system used in this experiment is able to analyze the video feed at around 20 frames per second, which takes about 5 predictions per one measurement update in the Kalman filter. However, by using the Kalman filter, the implementation usually requires good understanding of the delays present in the filter loop. In our case these cannot be determined in advance as there is a variable delay present in the system in the range from 150 to 300 ms, which is about 15–30 prediction steps of the Kalman filter. As the delay of the visual recognition system could not be predicted, we used the novel approach of 'timestamping' every

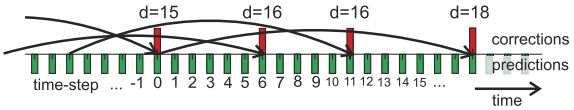


Fig. 8. A timeline of Kalman predictions and corrections.

frame produced by the camera with the current value of the on-board timer. The timer value is extracted from each captured frame and included in the visual recognition data packet that is sent back from the personal computer to the quadrocopter. There, the delay d is estimated by comparing the current value of the on-board timer with the value received in the data packet.

To compensate for the delay d of the received measurement $\mathbf{y}(k-d)$, step 4 of the Kalman-filtering algorithm is slightly modified

$$\hat{\mathbf{x}}(k) = \mathbf{x}^*(k) + \mathbf{K}'[\mathbf{y}(k) - \mathbf{C}\mathbf{x}^*(k-d)] \tag{35}$$

where $\mathbf{K}'$ will be discussed later. Note that a copy of Eq. (35) is needed for each of the three axes.

This is illustrated in Fig. 8. The video-camera frame, captured at the time-step 0, arrives into the Kalman correction step at the time-step 18, when the delay is estimated (in this case the delay is 18 samples). In that time-step, the received measurement $\mathbf{y}(18)$ is compared with the prediction output $\mathbf{x}^*(18-18)$ ($k=0$ is the time when the video-camera frame was captured). As can also be seen in Fig. 8, the predictions are executed at a regular rate of 100 predictions per second, while the corrections are executed at a much lower rate with irregular intervals. To accommodate these irregularities, the correction-time instants are defined by the variable $c(k)$:

$$c(k) = \begin{cases} 1, & \text{if new camera data is available at time-step } k \\ 0, & \text{otherwise.} \end{cases} \tag{36}$$

With this approach, the delay has been effectively transferred to the Kalman-filter correction loop and as the paper [26] suggests, the pre-calculated Kalman-filter gain has to be adjusted to ensure the filter's optimality. The modified Kalman gain $\mathbf{K}'$ can be calculated as in

$$\mathbf{K}' = \left[\prod_{i=0}^{d-1}(\mathbf{I} - \mathbf{K}(k-i)\mathbf{C}(k-i))\mathbf{A}(k-i-1)\right]\mathbf{K}(k) \tag{37}$$

which in our case only depends on the delay d and therefore $\mathbf{A}_1 = \mathbf{A}_2 \cdots = \mathbf{A}_k, \mathbf{C}_1 = \mathbf{C}_2 \cdots = \mathbf{C}_k$. As the correction step of the Kalman filter is executed only in time-steps in which the delayed measurement data $\mathbf{y}(k)$ is available, the following can be defined

$$\mathbf{K}(k) = \begin{cases} \overline{\mathbf{K}}, & c(k) = 1 \\ \mathbf{0}, & c(k) = 0. \end{cases} \tag{38}$$

With definition (38) the filter with discontinuous executions of the correction steps can be handled as a generic Kalman filter. Half-duplex communication, the usage of a non-real-time operating system and various other effects impact on the amount of delay of the visual system (the delay between the moment when the measurement of the state $\mathbf{x}(k-d)$ is produced and the corresponding $\mathbf{y}(k)$ is received). On the other hand, the video camera captures frames regularly every 5 time-steps. This means that during the delay of the visual system, 3–6 visual system results are received by the quadrocopter and thus the same number of Kalman-filter corrections was executed (illustrated in Fig. 8).

Although, the exact delay was used earlier to properly temporally align the past predicted states and the measurements, the experiments showed that using the delay to fine-tune the Kalman filter gain in each correction step had a minor effect on the system performance. During the experiments, it was determined

Table 1
Fixed Kalman gains for all modes.

	Mode 0	Mode 1	Mode 2
$\mathbf{K}'_x$	/	$\begin{bmatrix} 0.0097 \\ 0.0644 \\ 0.0063 \end{bmatrix}$	$\begin{bmatrix} 0.0820 & 0.0023 \\ 0.0022 & 0.0641 \\ 0.0002 & 0.0063 \end{bmatrix}$
$\mathbf{K}'_y$	/	$\begin{bmatrix} 0.0097 \\ 0.0644 \\ 0.0063 \end{bmatrix}$	$\begin{bmatrix} 0.0820 & 0.0023 \\ 0.0022 & 0.0641 \\ 0.0002 & 0.0063 \end{bmatrix}$
$\mathbf{K}'_z$	/	/	$\begin{bmatrix} 0.0852 \\ 0.0834 \\ 0.0075 \end{bmatrix}$

that, on average, $\rho = 4$ corrections are made per delay, so the expression (37) was used to define the expressions that produce the fixed Kalman gains $\mathbf{K}'$ for each mode of the Kalman filter, except the gains for the z-axis velocity measurements, as these are not delayed (they come from the air-pressure sensor)

$$\mathbf{K}' = \left[(\mathbf{I} - \overline{\mathbf{K}}\mathbf{C}_0)\mathbf{A}_0\right]^{\rho}\overline{\mathbf{K}}. \tag{39}$$

The Kalman gains, defined in Table 1, were hard-coded into the firmware. Note that the Kalman gains in Table 1 are used for filtering the measurements coming from the visual system. In mode 2 the positions are measured for all three axes, while the velocities are measured in mode 1 and 2 for the x- and y-axis.

5. Controller design

Because of its simple structure and robustness, a state-space controller was selected for the x- and y-axis, and because of the strong integrating part, required for the z-axis controller, a PI-D controller was selected for the z-axis. A problem we encountered during the controller design phase, i.e., that the manufacturer of the quadrocopter does not provide any information on the internal, primary, closed-loop controller that is used to follow the tilt and thrust commands. Parametric identification was used to approximate the quadrocopter's internal transfer functions and use them to design the controllers in the outer loop.

5.1. Parametric identification

For the purpose of parametric identification a human pilot was instructed to fly the quadrocopter above the target so that the visual recognition system was in mode 2 during the experiment. The pilot's control commands for the thrust, pitch and roll were recorded together with the responses of the quadrocopter—pitch and roll angles, vertical acceleration, the velocity and the positions in all three axes. The quadrocopter system transfer function from the control input to the quadrocopter's position was separated into three first-order transfer functions per axis (illustrated in Fig. 9) and the results are as follows:

1. Transfer functions in x- and y-axis are the same:

$$G_{1,xy}(z) = \frac{1.1}{z-0.95} \qquad G_{2,xy}(z) = \frac{0.00015}{z-0.9991}$$
$$G_{3,xy}(z) = \frac{0.01}{z-1}. \tag{40}$$

2. Transfer functions for the z-axis:

$$G_{1,z}(z) = \frac{0.0066}{z-0.853} \qquad G_{2,z}(z) = \frac{0.0811}{z-0.997}$$
$$G_{3,z}(z) = \frac{0.01}{z-1}. \tag{41}$$

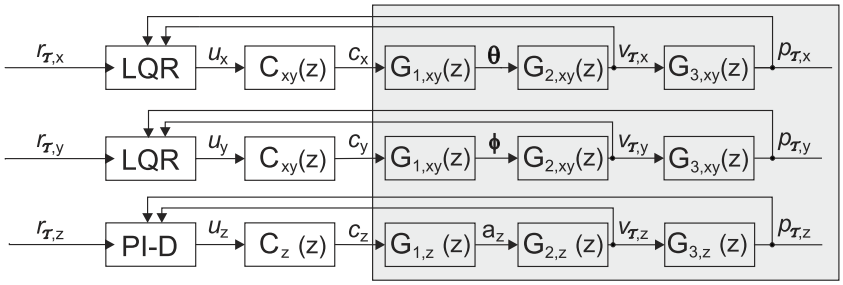


Fig. 9. Overview of the system.

5.2. Compensator design

In order to improve the dynamics of the closed-loop system, we decided to include a lead compensator (as illustrated in Fig. 9). The zeros of the compensators were placed at $z_{xy} = 0.95$ for the x and y-axes and at $z_z = 0.85$ for the z-axis, respectively. Similarly, the poles of the compensators were placed at $p_{xy} = 0.85$ and at $p_z = 0.75$. The gain of the compensator was set so that the gain of the compensator and the transfer function $G_1(z)$ combined was equal to 1.

$$C_{xy}(z) = 0.968\frac{z-0.95}{z-0.85} \qquad C_z(z) = 37.5\frac{z-0.85}{z-0.75}. \tag{42}$$

5.3. The controller for the x and the y-axis

The included compensator effectively compensated the dynamic effects of the $G_{1,xy}(z)$ transfer function. The remaining two transfer functions defined in Eq. (40) were then used to design a Linear Quadratic Regulator (LQR) [37] for the control of the quadrocopter in the x and y-axes. The transfer functions were transformed into the state-space form (43), (44) in such a way that the physical meaning of the states remained—one state for the velocity and the other for the position.

$$\mathbf{x}_x^{\mathrm{LQR}}(k+1) = \begin{bmatrix} 0.9991 & 0 \\ 0.0012 & 1 \end{bmatrix} \mathbf{x}_x^{\mathrm{LQR}}(k) + \begin{bmatrix} 0.0156 \\ 0 \end{bmatrix} u_x^{\mathrm{LQR}}(k) \tag{43}$$

$$y_x^{\mathrm{LQR}}(k) = \begin{bmatrix} 0 & 0.08 \end{bmatrix} \mathbf{x}_{\mathrm{LQR}}(k). \tag{44}$$

Knowing that the system in Eq. (43) is stabilizable, the design criterion for the best control gain $\mathbf{K}_{\mathrm{LQR}}$ is set up accordingly:

$$J = \sum_{k=1}^{\infty} \left(\mathbf{x}_x^{\mathrm{LQR}}(k)^T \mathbf{Q} \mathbf{x}_x^{\mathrm{LQR}}(k) + \mathbf{u}_x^{\mathrm{LQR}}(k)^T \mathbf{R} \mathbf{u}_x^{\mathrm{LQR}}(k) \right) \tag{45}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are weighting matrices, which were selected as follows:

$$\mathbf{Q} = \begin{bmatrix} 150 & 0 \\ 0 & 50 \end{bmatrix} \qquad \mathbf{R} = [0.1]. \tag{46}$$

The calculated gain was

$$\mathbf{K}_{\mathrm{LQR}} = \begin{bmatrix} 28.7 & 16.6 \end{bmatrix}. \tag{47}$$

For each time step k, the control inputs $u_x(k)$ and $u_y(k)$ are calculated based on the position and velocity errors and the controller gain $\mathbf{K}_{\mathrm{LQR}}$.

$$u_x(k) = \mathbf{K}_{\mathrm{LQR}} \begin{bmatrix} -v_{\mathcal{T},x} \\ e_x(k) \end{bmatrix} \qquad u_y(k) = \mathbf{K}_{\mathrm{LQR}} \begin{bmatrix} -v_{\mathcal{T},y} \\ e_y(k) \end{bmatrix} \tag{48}$$

where

$$e_x(k) = r_{\mathcal{T},x} - p_{\mathcal{T},x} \qquad e_y(k) = r_{\mathcal{T},y} - p_{\mathcal{T},y}. \tag{49}$$

The $r_{\mathcal{T},x}$ and the $r_{\mathcal{T},x}$ are the references for the position in the direction of the x- and y-axis of the target coordinate system. The output of the controller is then fed to the compensator $C_{xy}(z)$ to produce the final controller outputs $c_x(k)$ and $c_y(k)$.

$$c_x(k) = 0.85c_x(k-1) + 0.968\,(u_x(k) - 0.95u_x(k-1)) \tag{50}$$

$$c_y(k) = 0.85c_y(k-1) + 0.968\left(u_y(k) - 0.95u_y(k-1)\right). \tag{51}$$

5.4. The controller for the z-axis

For the z-axis controller, a discrete PI-D controller[1] was selected with the control algorithm:

$$u_z(k) = K_{P,z}e_z(k) + K_{I,z}\sum_{i=1}^{k} e_z(i-1) + K_{D,z}(-v_{\mathcal{K},z}(k)) \tag{52}$$

where

$$e_z(k) = r_{\mathcal{T},z} - p_{\mathcal{T},z}. \tag{53}$$

The $r_{\mathcal{T},z}$ is the reference for the position in the direction of the z-axis of the target coordinate system. Like with the controllers for the x- and y-axis, the output of the controller is fed to the compensator and its output is used to control the thrust.

$$c_z(k) = 0.75c_z(k-1) + 37.5\,(u_z(k) - 0.85u_z(k-1)). \tag{54}$$

The initial parameters of the controller were selected based on the identified transfer functions, but they were later fine-tuned during the experiments to the following values:

$$K_P = 100 \qquad K_I = 7 \qquad K_D = 50. \tag{55}$$

The parameters of the controller (55) are tuned for a particular operating point. It would be possible to schedule the parameters according to some signal, e.g. fuzzy gain scheduling could easily be implemented as reported by many applications in [38]. The results of experiments have shown that the system behaves quite robustly with fixed parameters (55). Also, since the controller design was not the primary objective of the paper, the parameters are fixed in our approach.

6. The experimental results

The complete system (illustrated in Fig. 2) was evaluated by two-part experiment. The first part of the experiment was focused on the quadrocopter responses during the reference-point changes. From the initial position at (0 cm, 0 cm, 70 cm) in the $\mathcal{T}$ coordinate system, the quadrocopter was instructed to move to the position (0 cm, −30 cm, 80 cm) and then return to the starting position. The move was made twice and the results are

[1] PI-D controller is a modification of the classical PID controller, where the derivative part of this PID controller is driven by the velocity state directly.

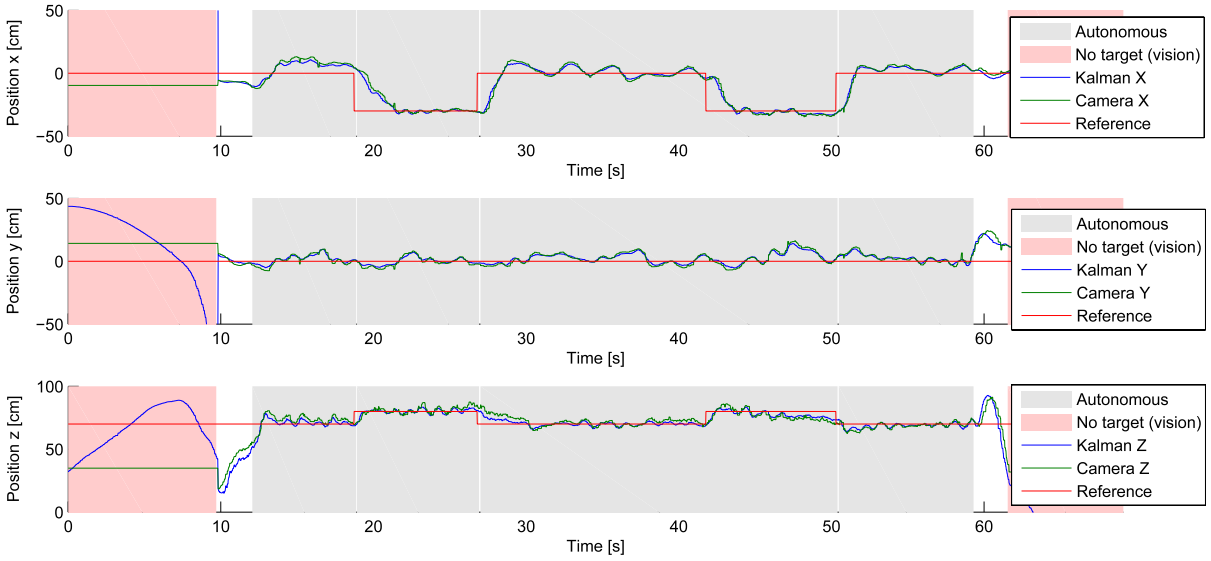


Fig. 10. Results of the experiment, where the responses to the reference-point changes were studied.

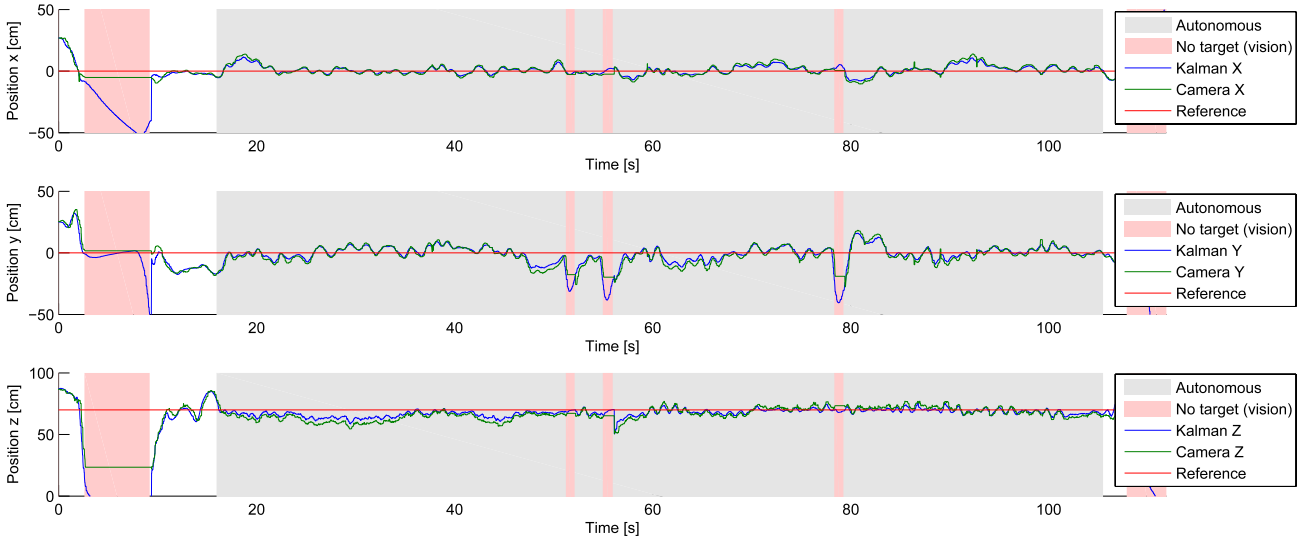


Fig. 11. Results of the experiment, where the responses to the external disturbances were studied.

shown in Fig. 10. The results indicate that the quadrocopter keeps its instructed position with minimal oscillations. Due to the small room where the tests were executed, a strong draft starts to form during the experiment. This draft results in a slight offset in the vertical position during the second half of the maneuvers.

The second part of the experiment mainly displays the quadrocopter robustness to a short-term loss of video-camera data. The quadrocopter is first put into autonomous hovering mode, in the moments $t_1 = 51$ s, $t_2 = 55$ s and $t_3 = 78$ s it is pushed out of the reference position by hand. Although the video-camera loses the target out of the view (illustrated by the red background in Fig. 11), the quadrocopter returns to the reference position and continues hovering.

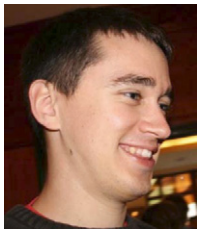
7. Conclusion

In this paper, the research results using Kalman filtering to fuse the highly-delayed, low-frequency, sensor measurements with no-delayed high-frequency measurements are presented. To accomplish this task some novel approaches were taken both in the visual recognition system and in the Kalman filtering. The solution was proven in multiple experiments where the quadrocopter autonomous hover mode was tested. The task was especially challenging as the solution is composed of software running on a personal computer, analyzing the video camera's data and logging the results, and microcontroller firmware running on-board the quadrocopter, taking care of the quadrocopter's inertial navigation, sensor fusion and control. The final system is semi-autonomous, as the visual recognition system is partly running off-board the quadrocopter.

References

[1] E. Altug, J. Ostrowski, R. Mahony, Control of a quadrotor helicopter using visual feedback, in: Proceedings of the IEEE International Conference on Robotics and Automation, IEEE, 2002, pp. 72–77.
[2] P. Zhang, J. Gu, E. Milios, P. Huynh, Navigation with IMU/GPS/digital compass with unscented Kalman filter, in: Proceedings of the IEEE International Conference on Mechatronics and Automation, vol. 3, IEEE, 2005, pp. 1497–1502.
[3] A. Kitanov, V. Tubin, I. Petrovic, Extending functionality of RF ultrasound positioning system with dead-reckoning to accurately determine mobile robot's orientation, in: Control Applications (CCA) & Intelligent Control (ISIC), IEEE, 2009, pp. 1152–1157.
[4] M. Achtelik, A. Bachrach, R. He, S. Prentice, N. Roy, Autonomous navigation and exploration of a quadrotor helicopter in GPS-denied indoor environments, in: Robotics: Science and Systems Conference, 2008.

[5] M. Blösch, S. Weiss, D. Scaramuzza, R. Siegwart, Vision based MAV navigation in unknown and unstructured environments, in: IEEE ICRA 2010, pp. 21–28.
[6] L. Teslić, I. Škrjanc, G. Klančar, Using a LRF sensor in the Kalman-filtering-based localization of a mobile robot, ISA Transactions 49 (2010) 145–153.
[7] K.E. Wenzel, P. Rosset, A. Zell, Low-cost visual tracking of a landing place and hovering flight control with a microcontroller, Journal of Intelligent and Robotic Systems 57 (2009) 297–311.
[8] N. Guenard, T. Hamel, R. Mahony, A practical visual servo control for an unmanned aerial vehicle, IEEE Transactions on Robotics 24 (2008) 331–340.
[9] D.-G. Sim, R.-H. Park, R.-C. Kim, S.U. Lee, I.-C. Kim, Integrated position estimation using aerial image sequences, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2002) 1–18.
[10] S. Hutchinson, G. Hager, P. Corke, A tutorial on visual servo control, IEEE Transactions on Robotics and Automation 12 (1996) 651–670.
[11] G. Chatterji, P. Menon, B. Sridhar, GPS/machine vision navigation system for aircraft, IEEE Transactions on Aerospace and Electronic Systems 33 (1997) 1012–1025.
[12] M. Tarhan, E. Altuğ, EKF based attitude estimation and stabilization of a quadrotor UAV using vanishing points in catadioptric images, Journal of Intelligent and Robotic Systems 62 (2011) 587–607.
[13] J. Kim, M.-S. Kang, S. Park, Accurate modeling and robust hovering control for a quadrotor VTOL aircraft, Journal of Intelligent and Robotic Systems 57 (2010) 9–26.
[14] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, D. Rus, Energy-efficient autonomous four-rotor flying robot controlled at 1 kHz, IEEE, April, 2007, pp. 361–366.
[15] S. Lupashin, A. Sch, M. Sherback, R.D. Andrea, A simple learning strategy for high-speed quadrocopter multi-flips, in: IEEE International Conference on Robotics and Automation, 2010, pp. 1642–1648.
[16] N. Michael, D. Mellinger, Q. Lindsey, V. Kumar, Experimental evaluation of multirobot aerial control algorithms, IEEE Robotics & Automation Magazine (2010) 56–65.
[17] O. Purwin, R. D'Andrea, Performing and extending aggressive maneuvers using iterative learning control, Robotics and Autonomous Systems 59 (2011) 1–11.
[18] S. Erhard, K.E. Wenzel, A. Zell, Flyphone: visual self-localisation using a mobile phone as onboard image processor on a quadrocopter, Journal of Intelligent and Robotic Systems 57 (2009) 451–465.
[19] Parrot, AR.Drone Parrot, 2011.
[20] T. Krajník, V. Vonásek, D. Fišer, J. Faigl, AR drone as a platform for robotic research and education, in: Research and Education in Robotics: EUROBOT 2011, Springer, Heidelberg, ISBN: 978-3-642-21974-0, 2011, pp. 172–186.
[21] L.R. García Carrillo, E. Rondon, A. Sanchez, A. Dzul, R. Lozano, Stabilization and trajectory tracking of a quad-rotor using vision, Journal of Intelligent and Robotic Systems 61 (2010) 103–118.
[22] S. Huh, D.H. Shim, A vision-based landing system for small unmanned aerial vehicles using an airbag, Control Engineering Practice 18 (2010) 812–823.
[23] S. Saripalli, J. Montgomery, G. Sukhatme, Vision-based autonomous landing of an unmanned aerial vehicle, in: Proceedings of the 2002 IEEE International Conference on Robotics and Automation, vol. 3, IEEE, 2002, pp. 2799–2804.
[24] F. Kendoul, I. Fantoni, K. Nonami, Optic flow-based vision system for autonomous 3D localization and control of small aerial vehicles, Robotics and Autonomous Systems 57 (2009) 591–602.
[25] H. Durrant-Whyte, T.C. Henderson, Multisensor data fusion, in: Springer Handbook of Robotics, 2001, pp. 585–610.
[26] T.D. Larsen, N.A. Andersen, O. Ravn, N.K. Poulsen, Incorporation of time delayed measurements in a discrete-time Kalman filter, in: Proceedings of the 37th IEEE Conference on Decision & Control, 1998, pp. 3972–3977.
[27] S. Lange, N. Sünderhauf, P. Protzel, A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments, in: International Conference on Advanced Robotics, ICAR 2009.
[28] A. Kirillov, AForge.NET: : Glyphs' recognition:, 2010.
[29] D.J. Fleet, Y. Weiss, Optical flow estimation, in: N. Paragios, Y. Chen, O. Faugeras (Eds.), Mathematical Models for Computer Vision: The Handbook, Springer, 2005, pp. 239–258.
[30] E. Rosten, T. Drummond, Machine learning for high-speed corner detection, in: European Conference on Computer Vision, 2006, pp. 430–443.
[31] B.D. Lucas, Generalized image matching by the method of differences, Ph.D. Thesis, Carnegie Mellon University, 1984.
[32] D.A. Forsyth, J. Ponce, Image formation: cameras, in: Computer Vision: A Modern Approach, 2003, pp. 3–27.
[33] G. Klancar, M. Kristan, R. Karba, Wide-angle camera distortions and non-uniform illumination in mobile robot tracking, Robotics and Autonomous Systems 46 (2) (2004) 125–133.
[34] M.S. Grewal, A.P. Andrews, General information, in: Kalman Filtering: Theory and Practice Using MATLAB, 2008, pp. 1–26.
[35] M.S. Grewal, A.P. Andrews, Linear optimal filters and predictors, in: Kalman Filtering: Theory and Practice Using MATLAB, 2008, pp. 131–178.
[36] A. Kelly, A 3D state space formulation of a navigation Kalman filter for autonomous vehicles, Carnegie Mellon University, May, 2006.
[37] G.F. Franklin, J.D. Powell, M.L. Workman, Multivariable and optimal control, in: Digital Control of Dynamic Systems, second ed., Addison-Wesley, 1990, pp. 417–480.
[38] R.-E. Precup, H. Hellendoorn, A survey on industrial applications of fuzzy control, Computers in Industry 62 (2011) 213–226.



Matevž Bošnak graduated in 2009 at the Faculty of Electrical Engineering, University of Ljubljana. He now works as a Ph.D. student in the Laboratory of Autonomous Mobile Systems at the same faculty.

Drago Matko received the B.Sc., M.Sc. and Ph.D. degrees in electrical engineering in 1971, 1973 and 1977 respectively, from the University of Ljubljana, Slovenia for work in the field of adaptive control systems. He visited the Institute of Space and Astronautical Science in Sagamihara Kanagawa, Japan, as a Foreign Research Fellow for 9 months in 1995–96 and for 6 months in 2003–04. He received an award from the Slovenian ministry for research and technology for his work in the field of computer-aided design of control systems in 1989 and the Zois award for achievements in science in 2003.

Sašo Blažič received the B.Sc., M.Sc., and Ph.D. degrees in 1996, 1999, and 2002, respectively, from the Faculty of Electrical Engineering, University of Ljubljana. Currently, he is an Associate Professor at the same faculty. The research interests of Sašo Blažič include the adaptive, fuzzy and predictive control of dynamical systems and the modeling of nonlinear systems. Lately, the focus of his research is in the area of autonomous mobile systems.