J Intell Robot Syst (2012) 67:43–60
DOI 10.1007/s10846-011-9646-5

# Quadrocopter Hovering Using Position-estimation Information from Inertial Sensors and a High-delay Video System

**Matevž Bošnak · Drago Matko · Sašo Blažič**

Received: 12 April 2011 / Accepted: 25 November 2011 / Published online: 10 December 2011


**Abstract** The requirement that mobile robots become independent of external sensors, such as GPS, and are able to navigate in an environment by themselves, means that designers have few alternative techniques available. An increasingly popular approach is to use computer vision as a source of information about the surroundings. This paper presents an implementation of computer vision to hold a quadrocopter aircraft in a stable hovering position using a low-cost, consumer-grade, video system. However, such a system is not able to stabilize the aircraft on its own and must rely on a data-fusion algorithm that uses additional measurements from on-board inertial sensors. Special techniques had to be implemented to compensate for the increased delay in the closed-loop system with the computer vision system, i.e., video timestamping to determine the exact delay of the vision system and a slight modification of the Kalman filter to account for this delay. At the end, the validation results of the proposed filtering technique are presented along with the results of an autonomous flight as a proof of the proposed concept.

M. Bošnak (✉) · D. Matko · S. Blažič
Faculty of Electrical Engineering,
University of Ljubljana, Tržaška cesta 25,
1000 Ljubljana, Slovenia
e-mail: matevz.bosnak@fe.uni-lj.si

**Keywords** Quadrocopter · Unmanned aerial vehicle (UAV) · Position estimation · Computer vision · Delayed measurements · Low-cost

## 1 Introduction

A quadrocopter 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 in 1921. However, quadrocopters are dynamically unstable and therefore suitable control methods are needed to make them stable [3]. These control methods are usually separated into two control loops—the high-speed, inner loop that controls the quadrocopter's attitude based on the outputs from the IMU (Inertial Measurement Unit) in a *strap-down* configuration [43] and the slower, outer loop that controls the quadrocopter's position. While the attitude of the quadrocopter can easily be determined by measuring the acceleration due to the Earth's gravitational field and the rotational velocities, there are no universal global positioning systems available. Outdoors, the GPS (Global Positioning System) that relies on receiving its signal from satellites can be used, however, indoors, the quadrocopter must rely on its own sensors to determine its position in the environment or on a custom local indoor tracking system, that must be built. Indoor tracking

systems are usually based either on beacon-based systems that use RF (Radio Frequency) waves (or a combination of RF and Ultrasonic waves [28]) or on an optical motion tracker (a constellation of video cameras with corresponding markers fitted to the quadrocopter) as in [3, 20, 25, 27, 34, 35, 37]. However, relying only on its own sensors and features in the environment makes a quadrocopter autonomous.

Most basic method to detect the movement of the quadrocopter relative to the ground below is to use a down-facing video camera and an optical-flow algorithm that compares two consequent captured video frames and extracts the relative movement of the visible features as was demonstrated in [16, 18, 24] and even in commercial product AR.Drone [1]. By summing together the relative optical-flow movements, this visual odometry data can be used for simple but not fully reliable positioning [10, 32]. Simplest upgrade for better positioning solution is to use the features of a deterministic environment—in [38, 39] authors used a helipad and in [23] authors used a runway as a feature, in [12, 32] authors used small circular landmarks, in [16, 19] authors used a four-point landing pad marker, in [41] authors used a Moir'e pattern and in [42] IR LEDs were used as a target feature. However, non-deterministic environments require the quadrocopter positioning system to build the map of the surroundings by themselves. Usually, Simultaneous localization and mapping (SLAM) is used to produce both the information on the location of the primary sensor and the map of the environment simultaneously. The SLAM algorithms are widely used in ground robotics [4, 40], but with the optimizations and improved mobile computing power they are more frequently used also in the Unmanned Aerial Vehicles (UAVs) [2, 5] and video gaming [22, 30]. Our approach uses the artificial fiducial marker (illustrated in Fig. 2), placed at the known position in the environment, removing the need of increased computing power of SLAM algorithms. Glyph features have the advantage that additional information (marker identification, its position, etc.) can be encoded into the feature itself.

Controlling objects using the data from these video-camera sensors is known as visual servo control. Visual servo control algorithms have become widely popular and used in the robotics and automation field in the past few decades. Most visual servo control systems were initially developed for serial-link robotic manipulators with the video camera typically mounted on the end-effector [21]. But due to the small sizes and weight of these sensors they are infiltrating the field of ground and aerial mobile robotics and even into the field of visual landing-assist technologies for airplanes [9].

The standard approach to visual servo control uses image-based measurements that enter the control loop. There are two similar approaches—a position-based visual servoing (PBVS) involves the extraction and reconstruction of the target pose with respect to the object using a camera and results in a Cartesian motion-planning problem, while the image-based visual servoing (IBVS) treats the problem as one of controlling features in the image plane and tries to align the extracted features with the reference ones [19]. The PBVS approach requires an accurate 3D model of the target, it is sensitive to video-camera calibration errors and it displays a tendency for the image features to leave the video camera's field of view and therefore lose the pose estimation. An image-based visual servoing, on the other hand, is less sensitive to video-camera non-linearities. However, PBVS approach enables navigation in the global coordinates which can be directly complemented with other positioning systems available.

Due to the limited amount of the processing power available on-board the small aerial vehicles, majority of solutions require the image processing algorithms to run on the ground station and are therefore be physically decoupled from the vehicle. This configuration imposes new challenges on how to robustly design the information flow between the ground station and the vehicle. The straight-forward solution is to use wired communication between the camera and the ground station (as is presented in [5, 12]), while the other is a wireless transmission of both the video camera frames and communication data. The latter approach can be found in majority of the papers dealing with the quadrocopters in the configuration where the ground station computer also executes the control algorithms ([3, 6, 19, 20, 31, 34, 35, 41, 44] and others), which imposes a

concern regarding the robustness of such a system in the cases of the communication drop-outs and the computer exceeding the real-time execution slots. The solution to the problem is to move the control algorithms to the vehicle's computer, which receives only the measurement results, obtained via the wireless communication (presented in [2, 16, 24] and others). This however introduces additional delays that must be dealt with. This paper presents our approach to the problem of fusing the position measurements with a considerable delay for the purpose of controlling the position of the quadrocopter.

To solve the issues with the delay and lost data, sensor-fusion methods in combination with a Kalman-filtering technique can be used [11]. In [33] different methods to solve the problem of delayed measurements are described. One is a method of extrapolating delayed measurements to present time and still obtain the optimality of the filter, while the other suggests updating the covariance matrix and the state at a different time. Our method is similar to the latter and uses past information on 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 data received via the wireless link. The IMU unit measures the rotational velocities 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. The correction stage with the delay-cancelling technique is executed only when the new image-based position data is received. Because the output of the Kalman filter is calculated every 10 milliseconds in the on-board computer, this system is resilient to short (in the range of one second) communication interruptions.

The system was put into practice on a quadrocopter X-3D-BL that is used as a testbed for many experiments ([5, 13, 20, 31, 35, 35, 37, 42] and others) and the visual servo control was evaluated during the dynamic and stationary flight of the quadrocopter which was affected by the 150 to 300 ms delays in visual system. This paper presents a slightly modified approach to visual servo control that provides promising results even with the delays mentioned. 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 the delay approximation is then presented, which complements the initial Kalman filter.

## 2 Quadrocopter Dynamics

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. 1) is a standard, right-handed, Cartesian coordinate system that has the origin in the target's center, the principal $x$ and $y$ axes in the plane of the target and the

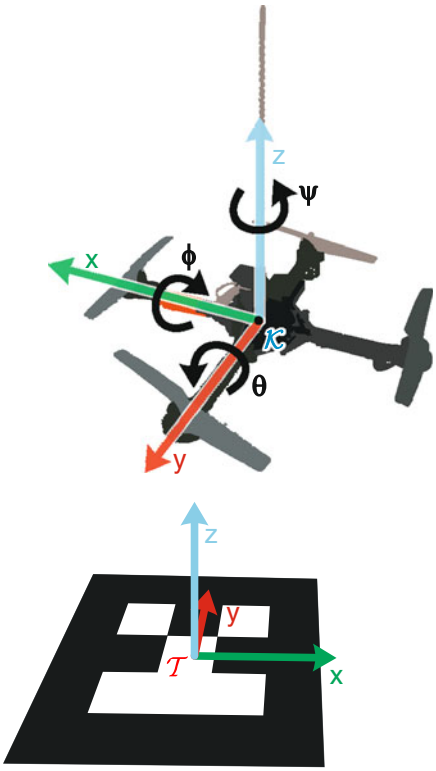


**Fig. 1** Graphical representation of the coordinate systems used

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. 1) has its origin in 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, 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.

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 high-level microprocessor has access to the attitude information of the low-level processor, the calibrated measurements of the acceleration, the gyroscope and 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 $\mathbf{p}_{\mathcal{T}} = (x, y, z)$ be the position of the centre 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

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

where $\mathbf{F}_{\mathrm{th},\mathcal{K}}$ is the total thrust vector in the quadrocopter coordinate system, $m$ is the mass of the quadrocopter treated as a solid body, $\mathbf{g}$ is the gravitational acceleration vector, $\mathbf{v}_{\mathcal{T}}$ 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

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

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

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

where $\mathbf{a}_{\mathcal{K}}$ is the acceleration vector in the quadrocopter's coordinate system $\mathcal{K}$ and $\mathbf{g}$ 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 sensors' output bias into account. The acceleration biases $\mathbf{a}_b$ (given in $\mathcal{K}$) must be estimated and then subtracted from the sensor reading $\mathbf{a}_s$ (also given in $\mathcal{K}$) as in

$$\frac{d^2}{dt^2}\mathbf{p}_{\mathcal{T}} = (\mathbf{a}_s - \mathbf{a}_b)\,\mathbf{R} - \mathbf{g} \qquad (4)$$

## 3 Computer Vision System

A computer vision system is one that combines both hardware and software with the aim to extract the information from images that is necessary to solve a specific task. In our case the vision system consists of a small, analogue, colour video camera with a wireless video transmitter (both fixed to the quadrocopter), a wireless video receiver, a computer with a video-capture card and

the software that was developed for this purpose. The whole vision system runs in real-time at 20 frames per second.

The vision system extracts the target's position and calculates the position of the origin and the orientation of $\mathcal{K}$ relative to $\mathcal{T}$. The software was developed in Visual C# using the AForge.NET framework for image processing.

### 3.1 Image Preparation

In the first step of the image processing, the received frame is deinterlaced to reduce the artefacts in the subsequent steps. Odd frames are doubled so that the produced frame has the same size as the original. Then, the image is transformed into a grayscale image to reduce the amount of data to process as only black and white colours are used for the target glyph.

### 3.2 Timestamp Extraction

For the purpose of the dynamic delay compensation, the value of the on-board, 8-bit timer is embedded into every frame as a timestamp (Fig. 2). This is done by an additional PIC microprocessor, which overlays the timer value on top of the video data in the fourth line of every video frame. On the ground computer the timestamp is extracted by analyzing only a strip of pixels in the timestamped line. The algorithm first searches for the darkest and the lightest pixels in the strip and these were used to determine the threshold that was used to binarize the pixel stripes, producing a series of white and black stripes of pixels, similar to a bar code, where the logical 0 is encoded with the larger ratio between black and white than logical 1. The combined length of white and black stripes for each bit is constant.

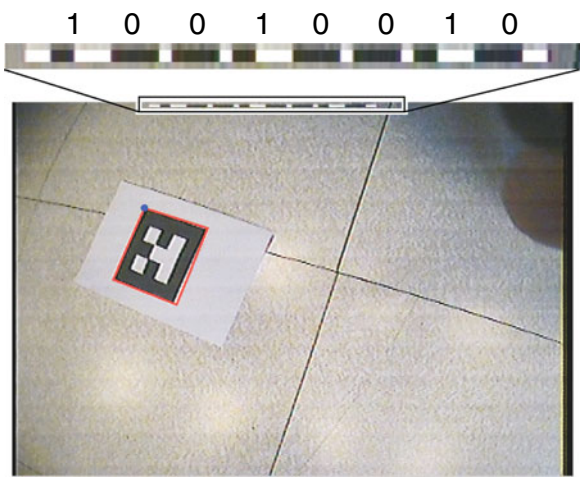


**Fig. 2** Extracting the timestamp from the captured image

### 3.3 Glyph Recognition

The glyph Recognition library GRATF is used for the glyph extraction. The target glyph (marked with a border in Fig. 2) is represented by a square grid divided equally into the same number of rows and columns. Each cell of the grid is filled with either a black or white colour. The first and the last row or column of each glyph contains only black cells, which creates a black border around each glyph. Also, we assume that every row and column has at least one white cell, so there are no completely black rows and columns (except the first and the last). A glyph is printed on white paper in such a way that there is a white area around the black borders of a glyph [26]. The glyph-extraction algorithm works on grayscale images and takes the following steps:

1. The edge-detection algorithm is executed on the grayscale image to search for the edges between the black and white borders.
2. The resulting image with edges is transformed into a black and white image by simple thresholding.
3. The blob-extraction algorithm is executed and all the connected blobs are identified.
4. The recognized blobs are filtered and checked to see if they form a quadrilateral shape.
5. The content of each blob is analyzed and checked with the glyph database to look for matches.

If there exists a match between the glyph found in the image and those in the database, the positions of the glyph's corner points are extracted and camera's viewpoint position is determined in the next step. Although the timestamp is located on top of the video frame, there exists a potential problem with a successful glyph recognition in cases when the glyph gets overlayed by the

timestamp or touches the edge of the video frame. Therefore the autonomous flight of the quadrocopter in positions, in which target glyph would touch timestamp or any edge of the video frame, is avoided.

### 3.4 Position Determination from the Image

The camera is mounted very close to the center of the quadrocopter's frame and thus we assume that the camera's viewpoint is at the origin of the $\mathcal{K}$ coordinate system. Also, we assume that the projective geometry of the pinhole camera is modelled by a perspective projection [14] with an additional radial distortion due to the wide-angle lens mounted on the camera. Wide-angle camera lenses are widely 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 [29] 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:

$$R = f(r) = \frac{H}{2}\frac{1 - e^{-2r/H}}{e^{-r/H}} = H \sinh \frac{r}{H} \tag{5}$$

where $R$ is the rectified radius, $r$ is the radius from the distorted image, defined in Eq. 6, and $H$ is the parameter of the lens, which is determined with the camera calibration. For a full-frame rectification, the above expression (5) must be evaluated for each pixel in the frame, which introduces increased computational load. However, the radial distortion does not greatly affect the glyph recognition algorithm and a full-frame radial distortion correction was not implemented. Instead, only the results of the glyph recognition (glyph corners' coordinates) are processed.

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 (5) is placed in the centre of the image. The relations between $(C_x, C_y)$ and $(C'_x, C'_y)$ are as follows:

$$\begin{aligned} r &= \sqrt{(C_x - C_x^p)^2 + (C_y - C_y^p)^2}, \\ \varphi &= \arctan2\left(C_y - C_y^p, C_x - C_x^p\right), \\ C'_x &= R\cos\varphi, \\ C'_y &= R\sin\varphi. \end{aligned} \tag{6}$$

where $(C_x^p, C_y^p)$ are the coordinates of the pixel in the center of the image and arctan2 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. 3 this point's projection onto the $x$–$z$ plane is illustrated), whose coordinates are expressed with respect to the $\mathcal{K}$ coordinate system, will project onto the image plane onto a point $C'_x, C'_y$, given by

$$\begin{bmatrix} C'_x \\ C'_y \end{bmatrix} = \frac{\lambda \nu}{T_z} \begin{bmatrix} T_x \\ T_y \end{bmatrix} \tag{7}$$

where $\lambda$ denotes the distance of the viewpoint origin behind the image plane [21], $\nu$ 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 $(\Delta T_x, \Delta T_y)$, that is compared to the size of the glyph in the image

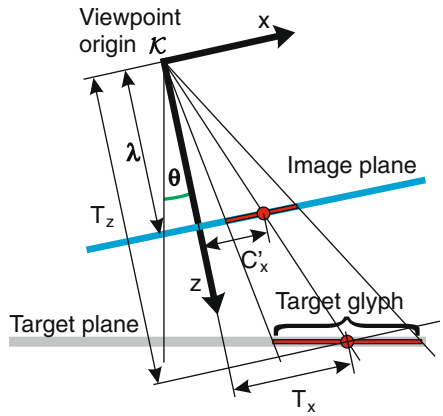


**Fig. 3** Calculating the position of the origin of the $\mathcal{K}$ in the $\mathcal{T}$ coordinate system

$(\Delta C_x, \Delta C_y)$, i.e., the distance between rectified glyph's corner points.

$$T_z = \lambda v \sqrt{\frac{\Delta T_x^2 + \Delta T_y^2}{\Delta C_x^2 + \Delta C_y^2}} = \lambda v \frac{a_T}{a_C} \tag{8}$$

where $a_T$ is the size of the target in millimetres and $a_C$ is the size of the image of the target in pixels. By combining Eqs. 7 and 8 the coordinates $T_x$ and $T_y$ can then be obtained

$$\begin{bmatrix} T_x \\ T_y \end{bmatrix} = \frac{a_T}{a_C} \begin{bmatrix} C'_x \\ C'_y \end{bmatrix} \tag{9}$$

The expressions (6) and (9) are evaluated for all four glyph's corner points, assuming that the ratio between target size $a_T$ and the distance from the camera's viewpoint to target $T_z$ is low and therefore all four distances between the camera's viewpoint and the glyph's corner points are the same.

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 $\psi$ 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.

$$\psi = \arctan 2 \frac{T_{y,0} - T_{y,2}}{T_{x,0} - T_{x,2}} \tag{10}$$

To determine the position of $\mathcal{K}$ with respect to $\mathcal{T}$ the effect of the video-camera tilt and orientation must be cancelled by rotating the point $\mathbf{T}$ about the x-axis for the angle $\phi$, about the y-axis for the angle $\theta$ and about the z-axis for the angle $\psi$.

$$\mathbf{T}^f(T_x^f, T_y^f, T_z^f) = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\psi) \begin{bmatrix} T_{x,c} \\ T_{y,c} \\ T_{z,c} \end{bmatrix} \tag{11}$$

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, and the coordinates $(T_{x,c}, T_{y,c}, T_{z,c})$ denote the center of the glyph (mean position of all four glyph's corner points). The rotation matrices are calculated based on the attitude data of the sensors. The coordinates $(-T_x^f, -T_y^f, -T_z^f)$ finally define the origin of $\mathcal{K}$ with respect to $\mathcal{T}$.

## 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. Therefore, an indirect visual servo control was developed that uses a combination of local position tracking using the integrated IMU unit, a dynamic delay estimation and compensation, an image-based position estimation and filtering with a Kalman filter (illustrated in Fig. 4).

As the quadrocopter, as a system, includes nonlinearities, it is common practice to employ the Extended Kalman filter (illustrated in Fig. 5), where a linear approximation is only used for solving the Riccati equation, a partial 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 [17]. 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 $a_K$ entering the Kalman filter (Fig. 6) in order to enable $a_T$ 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

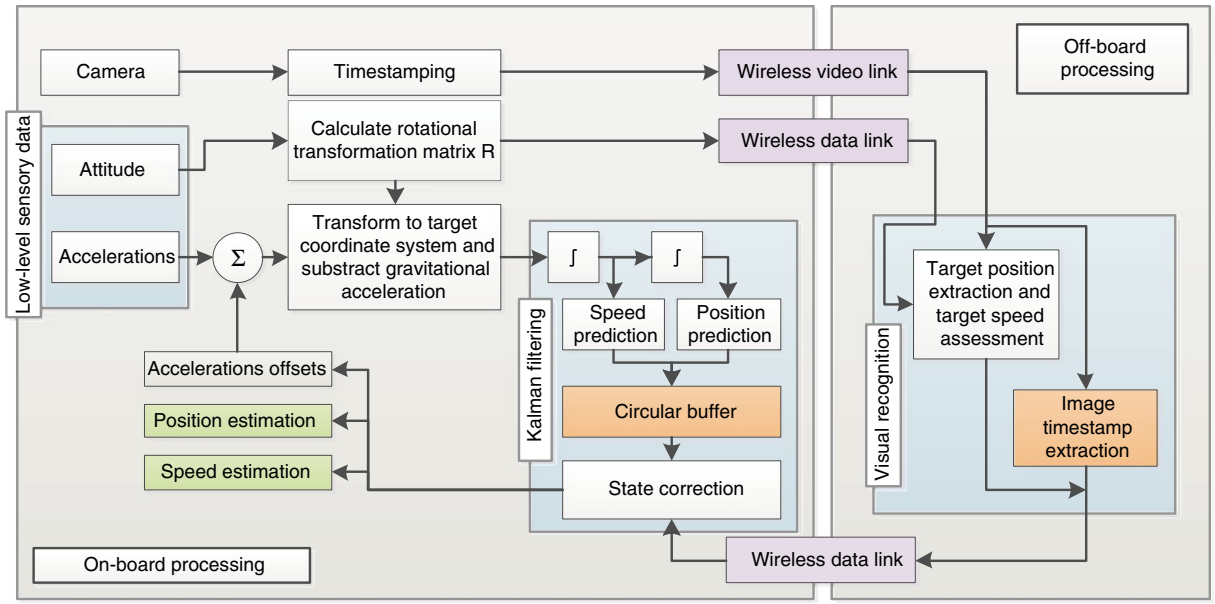


**Fig. 4** Block diagram of the system without controller

of the quadrocopter. Unfortunately, the inertial sensors measure the total acceleration, combining the effect of the dynamic acceleration due to 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 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 can be predicted and subtracted from the readings.

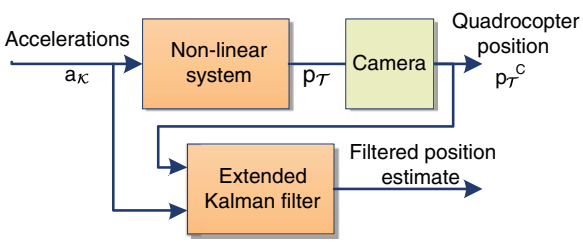


**Fig. 5** Block diagram of the system with the extended Kalman filter

Transforming 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 $\phi, \theta$ and $\psi$ 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

$$\mathbf{R}_{\mathrm{DCM}} = \begin{bmatrix} c\phi s\psi + c\psi s\phi s\theta & c\phi c\psi - s\psi s\phi s\theta & -c\theta s\phi \\ c\theta c\psi & -c\theta s\psi & s\theta \\ s\phi s\psi - c\phi s\theta c\psi & s\phi c\psi + c\phi s\theta s\psi & c\phi c\theta \end{bmatrix} \tag{12}$$

where these abbreviations were used

$$\begin{aligned} c\phi &= \cos\phi \quad c\theta = \cos\theta \quad c\psi = \cos\psi \\ s\phi &= \sin\phi \quad s\theta = \sin\theta \quad s\psi = \sin\psi \end{aligned} \tag{13}$$

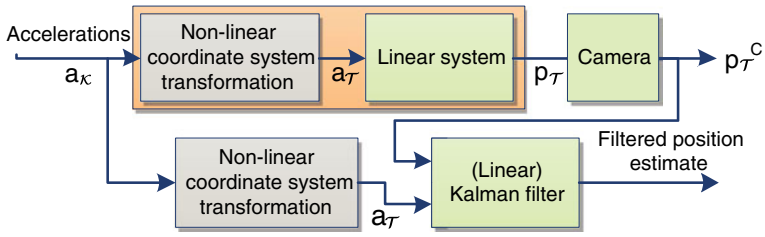


**Fig. 6** Block diagram of the system with the linear Kalman filter

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

$$\begin{aligned}\mathbf{a}_{d,\mathcal{T}}(k) &= \begin{bmatrix} a_{d,\mathcal{T},x}(k) \\ a_{d,\mathcal{T},y}(k) \\ a_{d,\mathcal{T},z}(k) \end{bmatrix} \\ &= \left( \left( \mathbf{a}_{s,\mathcal{K}}(k) - \mathbf{b}_{\mathcal{K}}(k) \right)^{T} \mathbf{R}_{\mathrm{DCM}}(k) - \begin{bmatrix} 0 & 0 & g \end{bmatrix} \right)^{T} \end{aligned} \tag{14}$$

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

$$\mathbf{a}_{s,\mathcal{K}}(k) = \begin{bmatrix} a_{s,\mathcal{K},x} \\ a_{s,\mathcal{K},y} \\ a_{s,\mathcal{K},z} \end{bmatrix} \qquad \mathbf{b}_{\mathcal{K}}(k) = \begin{bmatrix} b_{s,\mathcal{K},x} \\ b_{s,\mathcal{K},y} \\ b_{s,\mathcal{K},z} \end{bmatrix} \tag{15}$$

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

$$\mathbf{x}(k) = \begin{bmatrix} p_{\mathcal{T},x}(k) \\ v_{\mathcal{T},x}(k) \\ b_{\mathcal{K},x}(k) \\ p_{\mathcal{T},y}(k) \\ v_{\mathcal{T},y}(k) \\ b_{\mathcal{K},y}(k) \\ p_{\mathcal{T},z}(k) \\ v_{\mathcal{T},z}(k) \\ b_{\mathcal{K},z}(k) \end{bmatrix} \tag{16}$$

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, $b_{\mathcal{K},x}$, $b_{\mathcal{K},y}$ 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

$$\mathbf{x}(k+1) = \begin{bmatrix} \mathbf{A}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_0 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} \mathbf{B}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_0 \end{bmatrix} \mathbf{a}_{d,\mathcal{T}}(k) \tag{17}$$

$$\mathbf{y}(k) = \begin{bmatrix} \mathbf{C}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{C}_0 \end{bmatrix} \mathbf{x}(k) \tag{18}$$

where

$$\mathbf{A}_0 = \begin{bmatrix} 1 & T & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{B}_0 = \begin{bmatrix} \frac{T^2}{2} \\ T \\ 0 \end{bmatrix} \quad \mathbf{C}_0 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{19}$$

If Eq. 14 is inserted into Eq. 17, the following expression is produced

$$\begin{aligned}[b]\mathbf{x}(k+1) &= \left( \begin{bmatrix} \mathbf{A}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_0 \end{bmatrix} \mathbf{x}(k) - \begin{bmatrix} \mathbf{B}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_0 \end{bmatrix} \right. \\ &\quad \left. \times \mathbf{R}_{\mathrm{DCM}}^{T}(k) \mathbf{b}_{\mathcal{K}}(k) \right) \\ &\quad + \begin{bmatrix} \mathbf{B}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_0 \end{bmatrix} \left( \mathbf{R}_{\mathrm{DCM}}^{T}(k) \mathbf{a}_{s,\mathcal{K}}(k) - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \right) \end{aligned} \tag{20}$$

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 high delay caused by the image-capture and analysis procedure, the position-prediction stage exhibits a strong tendency to drift over time and the instantaneous reflection of the position changes of the quadrocopter. 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 the Eqs. 16, 17, 18, and 19, each of the axes can be inspected independently of each other. Here, we only show the solution for the x-axis. Identical solutions are applied to the other two axes. The state vector for the x-axis is defined as follows

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

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 [17]. This produces the following set of equations in the system space

$$\mathbf{x}_x(k+1) = \mathbf{A_0}\mathbf{x}_x(k) + \mathbf{B_0}a_{d,\mathcal{T},x}(k) + \mathbf{F}_x\mathbf{w}_x(k) \tag{22}$$

$$y_x(k) = \mathbf{C}_0\mathbf{x}_x(k) + n_x(k) \tag{23}$$

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

Usually, the Kalman filter is implemented in such a way that the following statements are evaluated:

1. At the time step $k$ a prediction is made for the time step $k+1$: $\mathbf{x}_x^*(k+1) = \mathbf{A}_0\hat{\mathbf{x}}_x(k) + \mathbf{B}_0 a_{d,\mathcal{T},x}(k)$.
2. Similarly, the prediction of the covariance matrix is produced for time $k+1$: $\mathbf{P}_x^*(k+1) = \mathbf{A}_0\hat{\mathbf{P}}_x(k)\mathbf{A}_0^T + \mathbf{F}_x\mathbf{V}_x\mathbf{F}_x^T$, where $\hat{\mathbf{P}}_x(k)$ is the estimation of the covariance matrix at the time step $k$, $\mathbf{V}_x$ is the covariance matrix of the process noise $\mathbf{w}_x$ and $\mathbf{F}_x$ is the input matrix of this noise.
3. In the new time step $k$, the Kalman gain matrix is produced first: $\mathbf{K}_x(k) = \mathbf{P}_x^*(k)\mathbf{C}_0^T[\mathbf{C}_0\mathbf{P}_x^*(k)\mathbf{C}_0^T + \mathbf{N}_x]^{-1}$, where $\mathbf{N}_x$ is the covariance matrix of the measurement noise $\mathbf{n}_x(k)$.
4. The estimation of the system state is then updated with the innovation: $\hat{\mathbf{x}}_x(k) = \mathbf{x}_x^*(k) + \mathbf{K}_x(k)[y_x(k) - \mathbf{C}_0\mathbf{x}_x^*(k)]$.
5. Finally, the estimation of the covariance matrix $\hat{\mathbf{P}}_x(k)$ is updated: $\hat{\mathbf{P}}_x(k) = \mathbf{P}_x^*(k) - \mathbf{K}_x(k)\mathbf{C}_0\mathbf{P}_x^*(k)$.

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 (20) 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 (20):

$$\begin{aligned} \frac{\partial \mathbf{x}_x(k+1)}{\partial \mathbf{x}_x(k)}\Big|_{\text{nom. op. point}} &= \mathbf{A}_0 - \mathbf{B}_0\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \\ &= \mathbf{A}_{L,0} = \begin{bmatrix} 1 & T & -\frac{T^2}{2} \\ 0 & 1 & -T \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{24}$$

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

By assuming that the process defined by Eqs. 22, 23 is not time-varying and the covariance matrices $\mathbf{N}_x$ and $\mathbf{V}_x$ are constant, the previous steps 2., 3. and 5. can be evaluated separately

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

There are several methods available for solving the steady-state matrix Riccati equation, the serial iteration being by far the most simple one on account of its slower convergence [17, 36]. The above three steps were iteratively executed in

Matlab until the relative change between two iterations was lower than $10^{-7}$. As the measurement vector has the form

$$y_x(k) = p_{\mathcal{T},x}(k-d) \tag{25}$$

where $d$ is a delay of the visual system (more description of the delay is given in Section 4.3), the following values for the process and measurement noise were used:

$$\mathbf{V}_0 = \begin{bmatrix} \sigma_{v,p} & 0 & 0 \\ 0 & \sigma_{v,v} & 0 \\ 0 & 0 & \sigma_{v,b} \end{bmatrix} \qquad \mathbf{N}_0 = \sigma_n^2 \tag{26}$$

where $\sigma_{v,p} = 0.1\ \text{cm}^2$, $\sigma_{v,v} = 0.1\ \text{cm}^2/\text{s}^2$, $\sigma_{v,b} = 0.01\ \text{cm}^2/\text{s}^4$, $\sigma_n = 2$ cm and $T_c = d_c T = 0.05$ s.

This produced the following Kalman gains

$$\mathbf{K} = \begin{bmatrix} \overline{\mathbf{K}}_0 \\ \overline{\mathbf{K}}_0 \\ \overline{\mathbf{K}}_0 \end{bmatrix}, \text{where } \overline{\mathbf{K}}_0 = \lim_{k\to\infty} \mathbf{K}_0(k) = \begin{bmatrix} 0.1552 \\ 0.1601 \\ 0.0145 \end{bmatrix} \tag{27}$$

Due to the assumption of $\mathbf{R}_{\text{DCM}}$ being a unitary matrix and incorporating the effect of sensor acceleration biases into the matrix $\mathbf{A}_{L,0}$ (24) 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 $\mathbf{R}_{\text{DCM}}$ 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 the implementation on the on-board, high-level processor as the number of operations over the matrices was greatly reduced and no matrix needed to be inverted. State prediction, as the first step in the filter, was already being made by the position-tracking system described in Section 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 to 30 prediction steps of the Kalman filter. As the delay of the visual recognition system could not be predicted, we used a novel approach of 'timestamping' every frame produced by the camera with the current value of the on-board timer. Later, the timestamped time is extracted (see Section 3.2) and included in the visual recognition data packet that is sent back from the personal computer to the quadrocopter (the whole system is illustrated in Fig. 4) where the delay $d$ is estimated by comparing the current value of the on-board timer with the timestamped time, 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

$$\hat{\mathbf{x}}(k) = \mathbf{x}^*(k) + \mathbf{K}'[\mathbf{y}(k) - \mathbf{C}\mathbf{x}^*(k-d)] \tag{28}$$

This is illustrated in Fig. 7. The video-camera frame, captured at the time-step 0, arrives into the Kalman correction step at the time-step 18, when

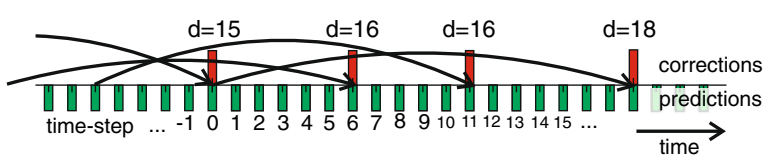


**Fig. 7** A timeline of Kalman predictions and corrections

the delay is estimated (in this case the delay is 18 samples). At 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). The delayed prediction output is fetched from the FIFO-style buffer, which was designed to store the prediction outputs for the delays of up to 256 time-steps.

As can also be seen in Fig. 7, 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 $c(k)$:

$$c(k) = \begin{cases} 1, \text{ if new camera data is available at} \\ \quad \text{time-step } k \\ 0, \text{ otherwise} \end{cases} \tag{29}$$

With this approach, the delay has been effectively transferred into the Kalman filter correction loop and as the paper [33] suggests, the precalculated Kalman filter gain has to be adjusted to ensure the filter's optimality. The modified Kalman gain $\mathbf{K}'$ can be calculated as in

$$\mathbf{K}' = \left[\prod_{i=0}^{d-1}(\mathbf{I} - \mathbf{K}(k-i)\mathbf{C}(k-i))\mathbf{A}(k-i-1)\right]\mathbf{K}(k) \tag{30}$$

which in our case only depends on the delay $d$ and therefore $\mathbf{A}_1 = \mathbf{A}_2 ... = \mathbf{A}_k$, $\mathbf{C}_1 = \mathbf{C}_2 ... = \mathbf{C}_k$. 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

$$\mathbf{K}(k) = \begin{cases} \mathbf{K}, & c(k) = 1 \\ \mathbf{0}, & c(k) = 0 \end{cases} \tag{31}$$

With definition (31) 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 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 to 6 visual system results are received by the quadrocopter and thus the same number of Kalman-filter corrections were executed (illustrated in Fig. 7).

At each correction step the gain of Kalman filter is recalculated using Eq. 30

$$\mathbf{K}_f = \left[(\mathbf{I} - \mathbf{KC})\mathbf{A}\right]^{\rho}\mathbf{K} \tag{32}$$

where $\rho$ is the delay expressed in number of Kalman gain corrections in the delay period and is obtained by the following expression

$$\rho = \sum_{i=0}^{d} c(k-i). \tag{33}$$

### 4.4 States Correction on Camera-image Re-acquisition

Using a Kalman filter when fusing the data from the IMU unit and the vision system, based on a camera with a limited view, resulted in a major disadvantage in transient responses at the moments when the target was re-acquired after leaving the video camera's field of view (left graphs in Fig. 8). In this case, many predictions are executed without the correction step of the Kalman filter and the state estimates start to drift. When the correction steps are executed again, relatively high Kalman gains cause unwanted oscillations of the filter estimate, which is not the result of the Kalman gain, defined in Eq. 32, as there are no measurements available when the target is out of the field of view of the camera and thus not related to the delay of the measurements. Even in a full Kalman-filter implementation, the covariance matrix $\mathbf{P}_0$ would get larger in each time-step in the absence of measurements and the resulting greater Kalman gain would produce even larger overshooting of the state estimates.

In these moments, vision-system measurements of the position have a much greater confidence level than the prediction outputs and therefore the

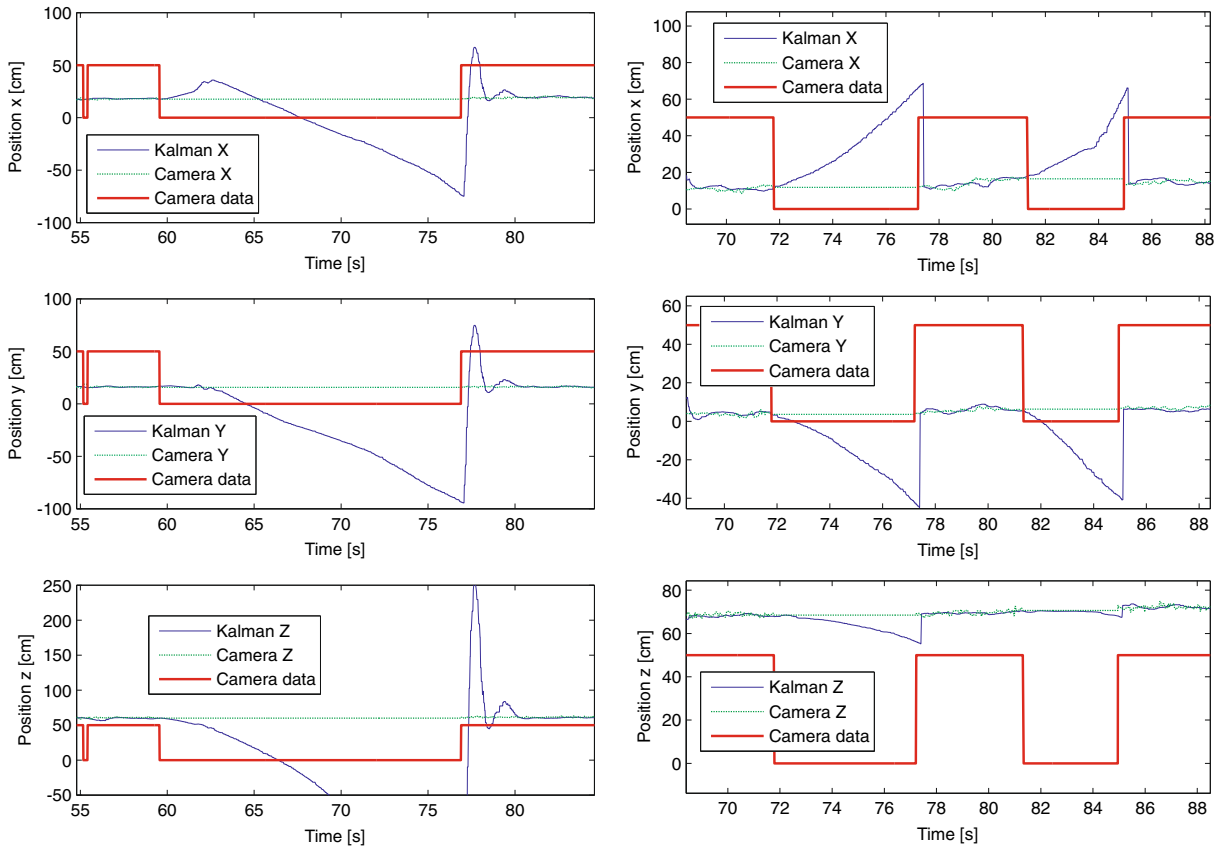


**Fig. 8** The effect of resetting the Kalman filter after a longer period of absence of the visual data. *Left* Filter outputs with no state resetting. *Right* Filter outputs with states resetting

states that reflect the quadrocopter's position can be directly updated with the help of the value of innovation, while the existing velocity states are used to predict the position from the time-step $k-d$ to $k$. The following resetting procedure is triggered when the vision-system measurements are absent for more than 100 predictions:

$$\begin{aligned}&\mathbf{x}_p(k-d) := \mathbf{y}(k)\\&\mathbf{x}_p(k-d+i) := \mathbf{x}_p(k-d+i) + (\mathbf{y}(k) - \mathbf{x}_p(k-d)))\\&\quad i = 1 \ldots d\end{aligned} \tag{34}$$

where $\mathbf{x}_p(k)$ is defined as the position subvector of the state vector $\mathbf{x}(k)$:

$$\mathbf{x}_p(k) = \begin{bmatrix} p_{\mathcal{T},x}(k) \\ p_{\mathcal{T},y}(k) \\ p_{\mathcal{T},z}(k) \end{bmatrix} \tag{35}$$

Experiments showed that this mechanism did not affect the Kalman filter during normal operation, but provided vital fast settling times at transient moments (right-hand graphs in Fig. 8, where the presence of video-camera data is shown as the value 50 and the absence of it with the value 0).

### 4.5 Experimental System

The quadrocopter as a system clearly has an integrating nature and as such requires position and velocity feedback controller loops for a stable hovering position. However, due to various external effects and a varying of the main-battery voltage, the quadrocopter's position exhibits a strong drift if the input signal is set to a constant value. This requires an additional integral controller to

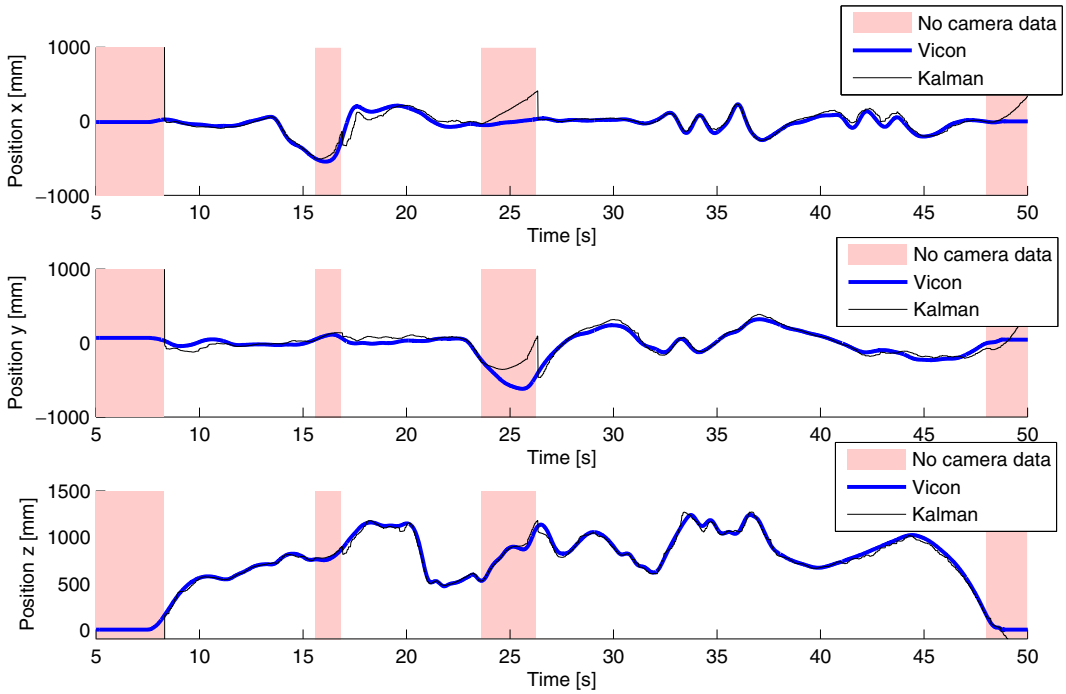


**Fig. 9** The validation of the Kalman-filter outputs with the Vicon measurements

be added to the state-feedback controller. This was accomplished by state augmentation [15] in order to include an additional integral state. A linear state-space model of the system (17), (18), (19) was augmented with the state $\chi_I$ (here shown for the x-axis only)

$$\chi_{I,x}(k+1) = \chi_{I,x}(k) + p_{\mathcal{T},x}(k) \tag{36}$$

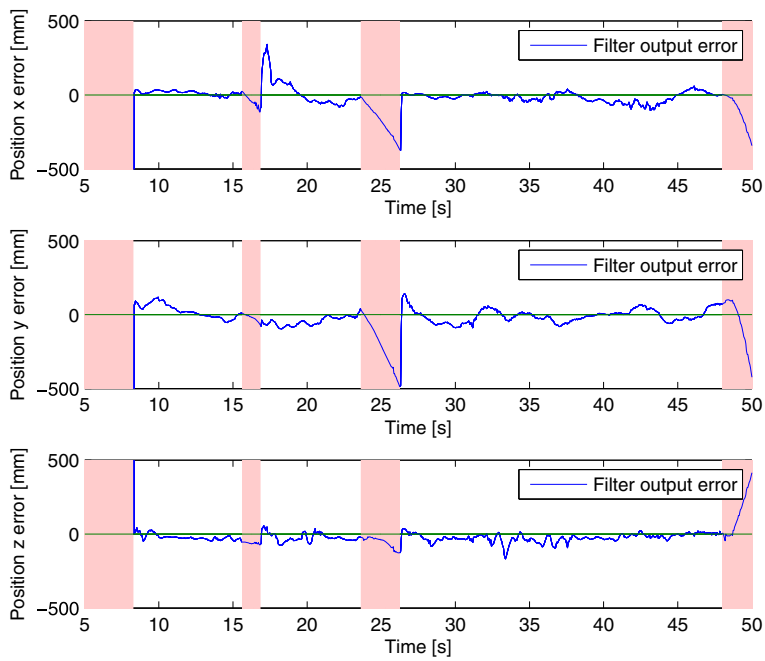


**Fig. 10** The position error of the proposed filtering system with respect to the Vicon measurements

The following control law was used

$$u(k) = -\begin{bmatrix} K_{\chi,x} & K_{p,x} & K_{v,x} \end{bmatrix} \begin{bmatrix} \chi_{I,x}(k+1) \\ p_{T,x}(k+1) \\ v_{T,x}(k+1) \end{bmatrix} \quad (37)$$

where the controller parameters were obtained by the pole placement method and rounded up to the following values:

$$\begin{bmatrix} K_{\chi,x} & K_{p,x} & K_{v,x} \end{bmatrix} = \begin{bmatrix} 1 & 50 & 45 \end{bmatrix}$$
$$\begin{bmatrix} K_{\chi,y} & K_{p,y} & K_{v,y} \end{bmatrix} = \begin{bmatrix} 1 & 50 & 45 \end{bmatrix} \quad (38)$$
$$\begin{bmatrix} K_{\chi,z} & K_{p,z} & K_{v,z} \end{bmatrix} = \begin{bmatrix} 8 & 95 & 40 \end{bmatrix}$$

## 5 Experimental Results

### 5.1 Visual System and Kalman Filtering Validation

The functions of both the visual system and the Kalman filtering were validated in the laboratory using the Vicon motion capture system that provides high frequency position measurements with sub-mm accuracy. For this purpose, six passive, infrared markers were attached to the quadrocopter and while the quadrocopter was flown manually above the target, the produced trajectory of all six markers and the estimations of the Kalman filter were recorded. From the positions of the markers, the position of the quadrocopter's center was calculated and compared to the Kalman-filter estimates of the position. The video of the experiment is available at [7]. The results, displayed in Figs. 9 and 10, show good matching for both measured trajectories. Between $t = 15.6$ s and $t = 16.8$ s and between $t = 23.6$ s and $t = 26.2$ s the target was out of camera's view and no position data from the visual system was present. In these periods, the filter output shows a drift of the position estimation.

### 5.2 Autonomous Flight

Experiments were made in the same laboratory environment as before. A human pilot was flying the quadrocopter to the vicinity of the target on

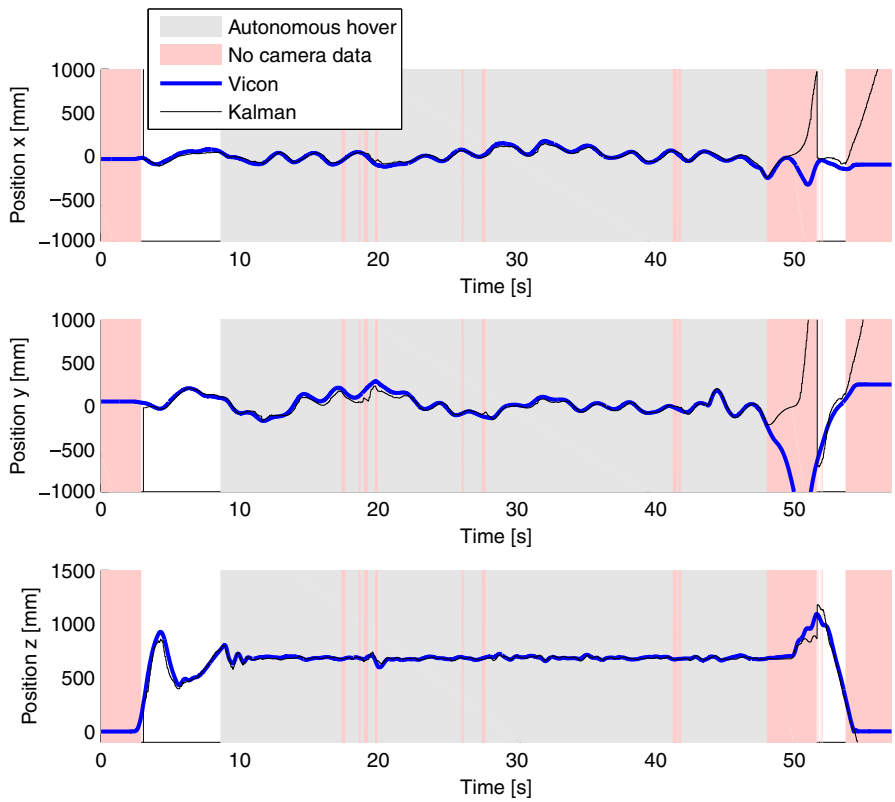


**Fig. 11** Quadrocopter's position in autonomous hovering mode

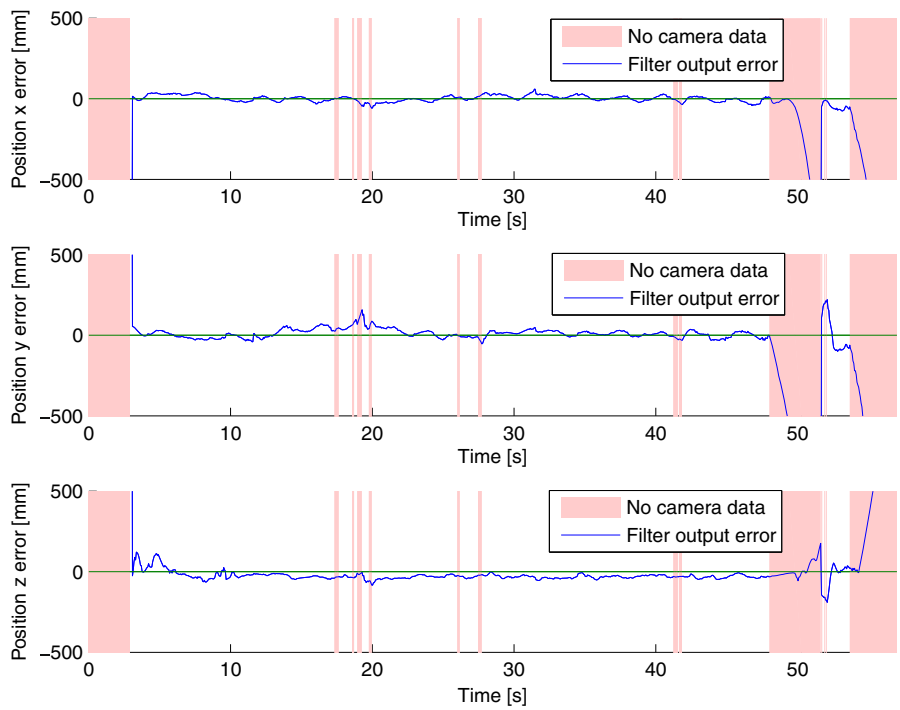


**Fig. 12** The position error of the proposed filtering system with respect to the Vicon measurements in autonomous hovering mode

the floor so that the target came into the field of view of the video camera. Then, a switch on the remote control was triggered causing the quadrocopter to begin hovering in autonomous mode. The reference position was a point 70 cm above the center of the target. The results are shown in Figs. 11 and 12 for each axis independently. The video of the experiment is available at [8].

At the beginning there is a short transient of the filter outputs due to the target appearing in the video camera's field of view. After that, the Kalman filter position output is 'locked' to the position of the quadrocopter above the target. During the hover, the camera was obstructed multiple times by waving a hand in front of camera (illustrated in Fig. 13), which is clearly noticeable by multiple vertical *No camera data* regions and most of times only slightly affected the autonomous hovering.

At $t = 44$ s, the quadrocopter was pushed slightly away from its reference position to demonstrate the response to the external disturbances. At $t = 47$ s, the quadrocopter was pulled away from the target and the camera lost the target from the field of view. The loss of visual system data lasted for more than a second and the on-board processor was not able to produce a valid prediction anymore. A human pilot had to

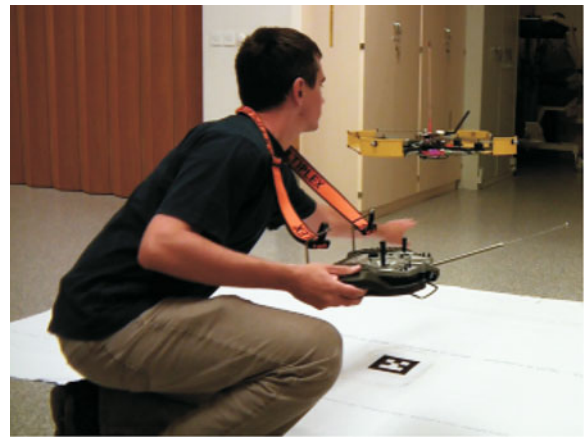

**Fig. 13** Obstructing the camera during the autonomous hover

take over the control of the quadrocopter and land it safely.

## 6 Conclusions and Future Work

### 6.1 Conclusions

The presented solution to the problem of fusing the variously delayed measurements in the Kalman filter is a result of beginning the research in this field. However, the early results look promising as a basis for advanced research and for the implementation of better control algorithms and an autonomous landing system. The experiments also show that the Extended Kalman filter for this system can successfully be avoided (as described in Section 4) with the aim to implement the system in a low-cost, embedded-microcontroller-based system.

### 6.2 Future Works

Future work will focus on the improvement of the control of the quadrocopter to increase its performance in both reference changes tracking and disturbance rejection. Both the video recognition and the control will be updated to allow the quadrocopter to perform autonomous take-off and landing manoeuvres. Focus will also be given on the updates that will allow the quadrocopter to fly outside the view of the primary landing target.

## References


1. AR.Drone Parrot: http://ardrone.parrot.com/. Accessed 26 Sept 2011
2. Achtelik, M., Bachrach, A., He, R., Prentice, S., Roy, N.: Autonomous navigation and exploration of a quadrotor helicopter in GPS-denied indoor environments. In: IEEE ICRA. http://iarc.angel-strike.com/oldauvs/5th_mission/2009SymposiumPapers/2009MIT.pdf (2009)
3. Altug, E., Ostrowski, J., Mahony, R.: Control of a quadrotor helicopter using visual feedback. In: Proceedings 2002 IEEE International Conference on Robotics and Automation, vol. 1, pp. 72–77. IEEE (2002)
4. Bailey, T., Durrant-Whyte, H.: Simultaneous localization and mapping (SLAM): part II. IEEE Robot. Autom. Mag. **13**(3), 108–117 (2006)
5. Blösch, M., Weiss, S., Scaramuzza, D., Siegwart, R.: Vision based MAV navigation in unknown and unstructured environments. In: IEEE ICRA 2010, pp. 21–28 (2010)
6. Bourquardez, O., Mahony, R., Guenard, N., Chaumette, F., Hamel, T., Eck, L.: Image-based visual servo control of the translation kinematics of a quadrotor aerial vehicle. IEEE Trans. Robot. (C) **25**(3), 2005–2008 (2009)
7. Bošnak, M.: Quadro: test 1. http://www.youtube.com/watch?v=0bpQeXlspAg
8. Bošnak, M.: Quadro: test 2. http://www.youtube.com/watch?v=xT-TxufjgVE
9. Chatterji, G., Menon, P., Sridhar, B.: GPS/machine vision navigation system for aircraft. IEEE Trans. Aerosp. Electron. Syst. **33**(3), 1012–1025 (1997)
10. Conte, G., Doherty, P.: Vision-based unmanned aerial vehicle navigation using geo-referenced information. EURASIP J. Adv. Signal Process. **2009**, 1–19 (2009)
11. Durrant-Whyte, H., Henderson, T.C.: Multisensor data fusion. In: Springer Handbook of Robotics, pp. 585–610 (2001)
12. Eberli, D., Scaramuzza, D., Weiss, S., Siegwart, R.: Vision based position control for MAVs using one single circular landmark. J. Intell. Robot. Syst. **61**(1–4), 495–512 (2011)
13. Erhard, S., Wenzel, K.E., Zell, A.: Flyphone: visual self-localisation using a mobile phone as onboard image processor on a quadrocopter. J. Intell. Robot. Syst. **57**(1–4), 451–465 (2009)
14. Forsyth, D.A., Ponce, J.: Computer Vision: a Modern Approach. Prentice Hall, Englewood Cliffs, NJ (2003)
15. Franklin, G., Powell, J., Workman, M.: Digital Control of Dynamic Systems. Prentice Hall, Englewood Cliffs, NJ (1998)
16. García Carrillo, L.R., Rondon, E., Sanchez, A., Dzul, A., Lozano, R.: Stabilization and trajectory tracking of a quad-rotor using vision. J. Intell. Robot. Syst. **61**(1–4), 103–118 (2010)
17. Grewal, M.S., Andrews, A.P.: Kalman filtering: theory and practice using MATLAB. Wiley-IEEE (2008)
18. Griffiths, S., Saunders, J., Curtis, A., Barber, B., McLain, T., Beard, R.: Maximizing miniature aerial vehicles. IEEE Robot. Autom. Magazine **13**(3), 34–43 (2006)
19. Guenard, N., Hamel, T., Mahony, R.: A practical visual servo control for an unmanned aerial vehicle. IEEE Trans. Robot. **24**(2), 331–340 (2008)
20. Gurdan, D., Stumpf, J., Achtelik, M., Doth, K.M., Hirzinger, G., Rus, D.: Energy-efficient autonomous four-rotor flying robot controlled at 1 kHz. In: Proceedings of the 2007 IEEE International Conference on Robotics and Automation, pp. 361–366 (2007)
21. Hutchinson, S., Hager, G., Corke, P.: A tutorial on visual servo control. IEEE Trans. Robot. Autom. **12**, 651–670 (1996)
22. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., Fitzgibbon, A.: KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In: Proceedings ACM User Interface and Software Technologies (ACM UIST), pp. 559–568 (2011)

23. Kaiser, M.K., Gans, N.R., Dixon, W.E.: Vision-based estimation for guidance, navigation, and control of an aerial vehicle. IEEE Transactions On Aerospace And Electronic Systems **46**(3), 1064–1077 (2010)
24. Kendoul, F., Fantoni, I., Nonami, K.: Optic flow-based vision system for autonomous 3D localization and control of small aerial vehicles. Robot. Auton. Syst. **57**(6–7), 591–602 (2009)
25. Kim, J., Kang, M.S., Park, S.: Accurate Modeling and Robust Hovering Control for a Quadrotor VTOL Aircraft. J. Intell. Robot. Syst. **57**(1–4), 9–26 (2009)
26. Kirillov, A.: AForge.NET: Glyphs' Recognition. http://www.aforgenet.com/articles/glyph_recognition/ (2010)
27. Kis, L., Prohaszka, Z., Regula, G.: Calibration and testing issues of the vision, inertial measurement and control system of an autonomous indoor quadrotor helicopter. In: Proceedings of the RAAD 2008 (2008)
28. Kitanov, A., Tubin, V., Petrovic, I.: Extending functionality of RF ultrasound positioning system with dead-reckoning to accurately determine mobile robot's orientation. In: Control Applications (CCA) & Intelligent Control (ISIC), pp. 1152–1157. IEEE (2009)
29. Klancar, G.: Wide-angle camera distortions and non-uniform illumination in mobile robot tracking. Robot. Auton. Syst. **46**(2), 125–133 (2004)
30. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, pp. 1–10. IEEE (2007)
31. Lange, S., Sünderhauf, N., Protzel, P.: Autonomous landing for a multirotor UAV using vision. In: International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR 2008), pp. 482–491 (2008)
32. Lange, S., Sünderhauf, N., Protzel, P.: 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 (2009)
33. Larsen, T.D., Andersen, N.A., Ravn, O., Poulsen, N.K.: Incorporation of time delayed measurements in a discrete-time Kalman filter. In: Conference on Decision & Control, pp. 3972–3977 (1998)
34. Liu, C., Chen, W.H., Andrews, J.: Piecewise constant model predictive control for autonomous helicopters. Robot. Auton. Syst. **59**(7–8), 571–579 (2011)
35. Lupashin, S., Sch, A., Sherback, M., Andrea, R.D.: A simple learning strategy for high-speed quadrocopter multi-flips. In: 2010 IEEE International Conference on Robotics and Automation, pp. 1642–1648 (2010)
36. Maciejowski, J.: Predictive Control with Constraints. Prentice-Hall, Englewood Cliffs, NJ (2002)
37. Michael, N., Mellinger, D., Lindsey, Q., Kumar, V.: The GRASP multiple micro-UAV testbed: experimental evaluation of multirobot aerial control algorithms. IEEE Robot. Autom. Mag. **17**(3), 56–65 (2010)
38. Saripalli, S., Montgomery, J., Sukhatme, G.: Vision-based autonomous landing of an unmanned aerial vehicle. In: Proceedings of the 2002 IEEE International Conference on Robotics and Automation, vol. 3, pp. 2799–2804. IEEE (2002)
39. Saripalli, S., Montgomery, J., Sukhatme, G.: Visually guided landing of an unmanned aerial vehicle. IEEE Trans. Robot. Autom. **19**(3), 371–380 (2003)
40. Teslić, L., Skrjanc, I., Klancar, G.: Using a LRF sensor in the Kalman-filtering-based localization of a mobile robot. ISA Transactions **49**(1), 145–153 (2010)
41. Tournier, G.P., Valenti, M., How, J.P., Feron, E.: Monocular vision and moiré patterns. In: AIAA (2006)
42. Wenzel, K.E., Rosset, P., Zell, A.: Low-cost visual tracking of a landing place and hovering flight control with a microcontroller. J. Intell. Robot. Syst. **57**(1–4), 297–311 (2009)
43. Zhang, P., Gu, J., Milios, E., Huynh, P.: Navigation with IMU/GPS/digital compass with unscented Kalman filter. In: Proceedings of the IEEE International Conference on Mechatronics and Automation, vol. 3, pp. 1497–1502. IEEE (2005)
44. Zhang, T., Kang, Y., Achtelik, M., Kuhnlenz, K., Buss, M.: Autonomous hovering of a vision/IMU guided quadrotor. In: 2009 International Conference on Mechatronics and Automation, pp. 2870–2875. IEEE (2009)