4.6 Concluding Remarks

In this chapter, an attitude control algorithm based on the motor armature current feedback was proposed and applied to a quad-rotor system. The control strategy, based on low-cost components, consists of adding an internal control loop on each ESC in such a way that for any given control input, the four motors turn at almost the same speed. The proposed controller was successfully tested in real-time experiments. The attitude stabilization performance has been considerably improved avoiding drift of the UAV from its desired angular position. In addition, robustness of the proposed controller with respect to external disturbances has been observed experimentally. Given that the quad-rotor Euler angles are very close to the origin, the resulting UAV can be effectively combined with other sensors, like GPS or imaging sensing systems, in order to perform autonomous positioning or trajectory tracking tasks.

Chapter 5
Imaging Sensors for State Estimation

The implementation of imaging systems onboard UAVs allows the development of original methodologies for extracting information concerning the surrounding environment, as well as of the vehicle itself. Imaging sensors, are very attractive since they are passive, non-contact, very versatile, and low-cost. In addition, they can be used in situations where other sensing devices fail, leading to a whole new group of applications that can be accomplished. However, before using imaging sensors a mathematical model must be computed, describing how the 3-dimensional points are represented in 2-dimensional images. For the interested reader, there are numerous computer vision books covering related topics in great detail, see for example [40] and [60]. In the present chapter the required theoretical background for the construction of imaging models is briefly presented, and in addition, the physical implementation of the imaging system for estimating the sates of a UAV is also addressed.

This chapter is divided as follows. Section 5.1 presents the pinhole camera model, as well as a camera calibration procedure for obtaining the intrinsic parameters. Next, stereo imaging is introduced in Sect. 5.2, with an explanation of the epipolar geometry concept. Also, a method for stereo calibration and rectification is presented. With the purpose of allowing the estimation of relative translational speed using an imaging sensor, the concept of optical flow and a method for its computation are detailed in Sect. 5.3. In Sect. 5.4, some important points that must be considered when implementing an imaging system onboard a quad-rotor UAV are discussed. In addition, the development of both a monocular and a stereo imaging system is presented, as well as the software architecture conceived with the purpose of estimating the data required for performing vision-based tasks. Finally, some concluding remarks are presented in Sect. 5.5.

5.1 Camera Model

Consider the central projection of 3-dimensional points in space onto a plane. Let the center of projection be the center of an Euclidean coordinate system, and consider

L.R. García Carrillo et al., *Quad Rotorcraft Control*, Advances in Industrial Control,
DOI 10.1007/978-1-4471-4399-4_5, © Springer-Verlag London 2013

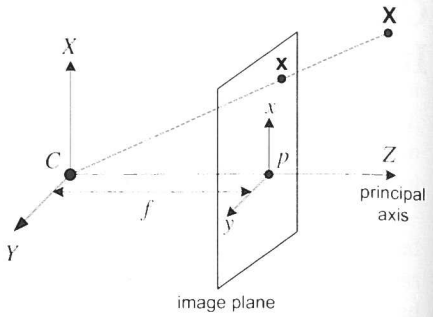


Fig. 5.1 Pinhole camera geometry. C is the camera center, p is the principal point. The camera center is placed at the coordinate origin. Note the image plane is placed in front of the camera center

the plane $Z = f$, which is known as the *image plane* or *focal plane*. Under the pinhole camera model, a point in space with coordinates $\mathbf{X} = (X, Y, Z)^{\mathrm{T}}$ is mapped to the point in the image plane where a line joining the point $\mathbf{X}$ to the center of projection meets the image plane, see Fig. 5.1. By similar triangles, it could be verified that the point $(X, Y, Z)^{\mathrm{T}}$ is mapped to the point $(f\frac{X}{Z}, f\frac{Y}{Z}, f)^{\mathrm{T}}$ in the image plane. Ignoring the image last coordinate, the central projection mapping from world to image coordinates is described as

$$(X, Y, Z)^{\mathrm{T}} \mapsto \left(f\frac{X}{Z}, f\frac{Y}{Z}\right)^{\mathrm{T}} \tag{5.1}$$

This is defined as a mapping from an 3-dimensional Euclidean space $\mathbb{R}^3$ to a 2-dimensional Euclidean space $\mathbb{R}^2$. The center of projection C is called the *camera center*, also known as the *optical center*. The line from the camera center perpendicular to the image plane is called the *principal axis* or *principal ray* of the camera, and the point where the principal axis meets the image plane is called the *principal point* and is represented by p. The plane through the camera center parallel to the image plane is called the *principal plane* of the camera [40].

Central Projection Using Homogeneous Coordinates If the world and image points are represented by homogeneous vectors, then central projection is simply expressed as a linear mapping between their homogeneous coordinates. In particular, equation (5.1) may be written in terms of matrix multiplication as

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5.2}$$

The 3×4 matrix in (5.2) may be written as $\mathrm{diag}(f, f, 1)[I|\mathbf{0}]$ where $\mathrm{diag}(f, f, 1)$ is a diagonal matrix and $[I|\mathbf{0}]$ represents a matrix divided up into a 3×3 block (the identity matrix) plus a column vector, here the zero vector. Let $\mathbf{X}$ be the notation for the world points represented by the homogeneous 4-dimensional vector $(X, Y, Z, 1)^{\mathrm{T}}$. Also, let $\mathbf{x}$ be the image points represented by a homogeneous 3-dimensional vector, and P for the 3×4 homogeneous *camera projection matrix*.

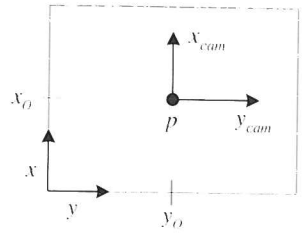


Fig. 5.2 Image (x, y) and camera $(x_{\mathrm{cam}}, y_{\mathrm{cam}})$ coordinates systems

Then, (5.2) is written compactly as

$$\mathbf{x} = P\mathbf{X} \tag{5.3}$$

Equation (5.3) defines the camera matrix for the pinhole model of central projection [40]

$$P = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.4}$$

Principal Point Offset Equation (5.1) assumes that the coordinates origin in the image plane is placed at the principal point. In practice, it may not be, so that in general there is a mapping

$$(X, Y, Z)^{\mathrm{T}} \mapsto \left(f\frac{X}{Z} + p_x, f\frac{Y}{Z} + p_y\right)^{\mathrm{T}} \tag{5.5}$$

where $(p_x, p_y)^{\mathrm{T}}$ are the coordinates of the principal point p, see Fig. 5.2. Expressing (5.5) in homogeneous coordinates one has

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{5.6}$$

Writing

$$K = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} \tag{5.7}$$

equation (5.6) has the form

$$\mathbf{x} = K[I|\mathbf{0}]\mathbf{q} \tag{5.8}$$

Matrix K is called the *camera calibration matrix*. In (5.8), $(X, Y, Z, 1)^{\mathrm{T}}$ has been written as $\mathbf{q}$ to emphasize that the camera is assumed to be located at the origin of a Euclidean coordinate system, with the principal axis of the camera pointing straight down the Z-axis, and the point $\mathbf{q}$ is expressed in this coordinate system. Such a coordinate system is called the *camera coordinate frame*.

Extrinsic Parameters In general, points in space will be expressed in terms of a different Euclidean coordinate frame, known as the *world coordinate frame*. The world and camera coordinate frames are related via a rotation and a translation. If $\tilde{\mathbf{X}} \in \mathbb{R}^{3\times 1}$ is an inhomogeneous vector representing the coordinates of a point in the world coordinate frame, and $\tilde{\mathbf{q}}$ represents the same point in the camera coordinate frame, then we may write $\tilde{\mathbf{q}} = R(\tilde{\mathbf{X}} - \tilde{\mathbf{C}})$, where $\tilde{\mathbf{C}}$ represents the coordinates of the camera center in the world coordinate frame, and $R \in \mathbb{R}^{3\times 3}$ is a rotation matrix representing the orientation of the camera coordinates frame. This equation can be written in homogeneous coordinates as

$$\mathbf{q} = \begin{bmatrix} R & -R\tilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} R & -R\tilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \mathbf{X} \tag{5.9}$$

Putting together (5.8) and (5.9) leads to

$$\mathbf{x} = KR[I| - \tilde{\mathbf{C}}]\mathbf{X} \tag{5.10}$$

where $\mathbf{X}$ is now in a world coordinate frame. This is the general mapping given by a pinhole camera. One sees that a general pinhole camera $P = KR[I| - \tilde{\mathbf{C}}]$, has nine degrees of freedom: three for K (the elements f, p_x, p_y), three for R, and three for $\tilde{\mathbf{C}}$. The parameters contained in K are called the *internal camera parameters*, or the *internal orientation* of the camera. The parameters of R and $\tilde{\mathbf{C}}$ which relate the camera orientation and position to a world coordinate system are called the *extrinsic parameters* or the *exterior orientation*. It is often convenient not to make the camera center explicit, and instead to represent the world to image transformation as $\tilde{\mathbf{q}} = R\tilde{\mathbf{X}} + \mathbf{t}$. In this case the camera matrix is simply

$$P = K[R|\mathbf{t}] \tag{5.11}$$

where from (5.10)

$$\mathbf{t} = -R\tilde{\mathbf{C}} \tag{5.12}$$

Intrinsic Properties The pinhole camera model just derived assumes that the image coordinates are Euclidean coordinates, having equal scales in both axial directions. In the case of charge-coupled device (CCD) cameras, there is the possibility of having non-square pixels. If image coordinates are measured in pixels, this has the extra effect of introducing unequal scale factors in each direction. In particular, if the number of pixels per unit distance in image coordinates are m_x and m_y in the x and y directions, respectively, then the transformation from world coordinates to pixel coordinates is obtained by multiplying (5.7) on the left by an extra factor $\text{diag}(m_x, m_y, 1)$. Thus, the general form of the calibration matrix of a CCD camera is

$$K = \begin{bmatrix} \alpha_x & & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix} \tag{5.13}$$

where $\alpha_x = fm_x$ and $\alpha_y = fm_y$ represent the focal length of the camera in terms of pixel dimensions in the x and y direction, respectively. Similarly, $\tilde{\mathbf{x}}_0 = (x_0, y_0)$ is the principal point in terms of pixel dimensions, with coordinates $x_0 = m_x p_x$ and $y_0 = m_y p_y$. A CCD camera thus has 10 degrees of freedom.

Consider now a camera calibration matrix K of the form

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix} \tag{5.14}$$

The added parameter s is referred to as the *skew parameter*, which will be zero for most normal cameras [40]. A camera

$$P = KR[I| - \tilde{\mathbf{C}}] \tag{5.15}$$

for which the calibration matrix is of the form in (5.14) is called a *finite projective camera*. A finite projective camera has 11 degrees of freedom. This is the same number of degrees of freedom as a 3×4 matrix, defined up to an arbitrary scale. The left hand 3×3 sub-matrix of P, equal to KR, is non-singular. Conversely, any 3×4 matrix P for which the left hand 3×3 sub-matrix is non-singular is the camera matrix of some finite projective camera, because P can be decomposed as $P = KR[I| - \tilde{\mathbf{C}}]$. Indeed, letting M be the left 3×3 sub-matrix of P, one decomposes M as a product $M = KR$ where K is upper-triangular of the form of (5.14), and R is a rotation matrix. This decomposition is essentially the RQ matrix decomposition. The camera matrix P can therefore be written $P = M[I|M^{-1}\mathbf{p}_4] = KR[I| - \tilde{\mathbf{C}}]$ where $\mathbf{p}_4$ is the last column of P.

The previous relationships fully describe how a general projective camera P maps 3-dimensional world points $\mathbf{X}$, to 2-dimensional image points $\mathbf{x}$, according to $\mathbf{x} = P\mathbf{X}$. However, cameras use lenses, which distort the images, more or less, depending on their design. In particular, lenses with a wide field of view tend to significantly distort images. Therefore, camera model has to be extended with a distortion model to compensate distortion before performing any image processing. For the distortion model, radial and tangential distortion must me assumed. Let $\mathbf{x}$ denote the projected coordinates of a point $\mathbf{X}$ before the multiplication with the camera calibration matrix K, and let $\mathbf{x}_d$ represent its corresponding distorted coordinates. The distortion model can be written as

$$\mathbf{x}_d = \underbrace{\mathbf{x}(1 + k_1 r^2 + k_2 r^4)}_{\text{radial}} + \underbrace{\begin{bmatrix} 2k_3 xy + k_4(r^2 + 2x^2) \\ k_3(r^2 + 2y^2) + 2k_4 xy \end{bmatrix}}_{\text{tangential}} \tag{5.16}$$

with $r^2 = x^2 + y^2$ and the distortion factor $L(d) = 1 + k_1 r + k_2 r^2 + k_3 r^3 + k_4 r^4$, considering the k_* factors as part of the interior calibration of the camera. The distortion model implies that nonlinear equations have to be solved in order to recover $\mathbf{x}$. Due to this, when image processing involves real-time applications, it is useful to pre-compute a look-up table that maps distorted to undistorted coordinates.

5.1.1 Camera Calibration

This section describes a method for estimating the camera projection matrix from corresponding 3-dimensional world and 2-dimensional image entities. The simplest such correspondence is that between a 3-dimensional point $\mathbf{X}$ and its corresponding 2-dimensional image $\mathbf{x}$ under the unknown camera mapping P. Given sufficiently many correspondences $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$, the camera matrix P may be determined. The internal parameters K of the camera may be extracted from the matrix P by means of an RQ decomposition.

Consider a number of point correspondences $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$ between 3-dimensional points $\mathbf{X}_i$ and 2-dimensional image points $\mathbf{x}_i$. A camera matrix P must be found, namely a 3×4 matrix such that $\mathbf{x}_i = P\mathbf{X}_i$ for all i. $\mathbf{x}_i = P\mathbf{X}_i$ involves homogeneous coordinates thus $\mathbf{x}_i$ and $P\mathbf{X}_i$ just have to be proportional (defined up to a scalar factor). Therefore, it is possible to use the cross product $\mathbf{x}_i \times P\mathbf{X}_i = 0$. Let $\mathbf{p}_1^{\mathrm{T}}$, $\mathbf{p}_2^{\mathrm{T}}$, $\mathbf{p}_3^{\mathrm{T}}$ be the three row vectors of P

$$P\mathbf{X}_i = \begin{bmatrix} \mathbf{p}_1^{\mathrm{T}}\mathbf{X}_i \\ \mathbf{p}_2^{\mathrm{T}}\mathbf{X}_i \\ \mathbf{p}_3^{\mathrm{T}}\mathbf{X}_i \end{bmatrix} \quad ; \quad \mathbf{x}_i \times P\mathbf{X}_i = \begin{bmatrix} y_i\mathbf{p}_3^{\mathrm{T}}\mathbf{X}_i - w_i\mathbf{p}_2^{\mathrm{T}}\mathbf{X}_i \\ w_i\mathbf{p}_1^{\mathrm{T}}\mathbf{X}_i - x_i\mathbf{p}_3^{\mathrm{T}}\mathbf{X}_i \\ x_i\mathbf{p}_2^{\mathrm{T}}\mathbf{X}_i - y_i\mathbf{p}_1^{\mathrm{T}}\mathbf{X}_i \end{bmatrix}$$

for each correspondence $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$ there exists a relationship

$$\begin{bmatrix} \mathbf{0}^{\mathrm{T}} & -w_i\mathbf{X}_i^{\mathrm{T}} & y_i\mathbf{X}_i^{\mathrm{T}} \\ w_i\mathbf{X}_i^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & -x_i\mathbf{X}_i^{\mathrm{T}} \\ -y_i\mathbf{X}_i^{\mathrm{T}} & x_i\mathbf{X}_i^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} \end{bmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{pmatrix} = 0 \tag{5.17}$$

where each $\mathbf{p}_i^{\mathrm{T}}$ is a 4-vector, the ith row of P. Since the three equations of (5.17) are linearly dependent, one may choose to use only the first two equations

$$\begin{bmatrix} \mathbf{0}^{\mathrm{T}} & -w_i\mathbf{X}_i^{\mathrm{T}} & y_i\mathbf{X}_i^{\mathrm{T}} \\ w_i\mathbf{X}_i^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & -x_i\mathbf{X}_i^{\mathrm{T}} \end{bmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{pmatrix} = 0 \tag{5.18}$$

One obtains two independent equations in 11 unknowns (ignoring scale). From a set of n point correspondences (6 as minimum), we obtain a $2n \times 12$ matrix A, by stacking up (5.18) for each correspondence. The projection matrix P is computed by solving the set of equations $A\mathbf{p} = 0$, where $\mathbf{p}$ is the vector containing the entries of the matrix P.

Calibration Using a Chessboard A common calibration method consists of placing in front of the camera an object with certain number of points whose coordinates are well known in a 3-dimensional reference frame. In principle, any appropriately characterized object could be used as a calibration object, yet the practical choice is a regular pattern such as a chessboard. Some calibration methods in the literature rely on 3-dimensional objects (e.g., a box covered with markers), but flat chessboard patterns are much easier to deal with than 3-dimensional calibration

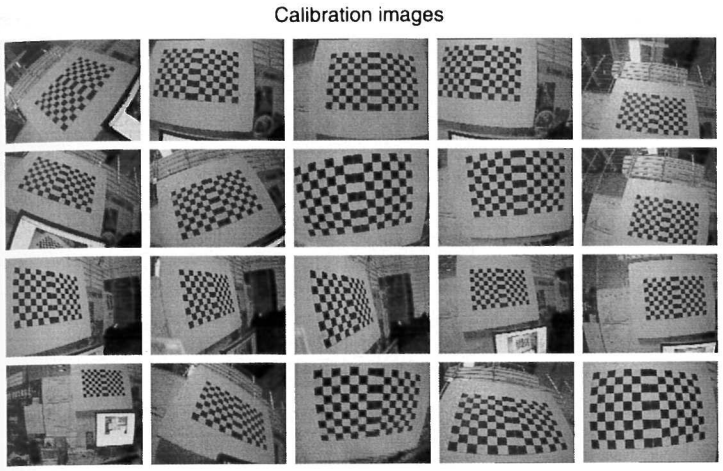


Fig. 5.3 The *Camera Calibration Toolbox for Matlab* GUI displaying the chessboard images

objects. For the previous reason, a chessboard pattern has been built, consisting of alternating black and white squares of 30 mm each.

A calibration technique that requires the camera to observe a planar chessboard shown at a few (at least two) different positions and orientations is presented in [83]. The algorithm takes advantage of the chessboard pattern to compute the square's corners points, then, it extracts the projective transformation between the image points of the n different images, up to a scale factor. With this method, the intrinsic and extrinsic parameters of the camera are computed, in addition, radial lens distortion is modeled as well. The proposed procedure consists of a closed-form solution, followed by a nonlinear refinement based on the maximum likelihood criterion.

A handy implementation of the previously presented calibration method is the *Camera Calibration Toolbox for Matlab* [17]. This program, available online for downloading and installing, has shown to be a very powerful and easy to use tool, capable of accurately compute the intrinsic and extrinsic parameters of a camera. For the present research, the *Camera Calibration Toolbox for Matlab* was used to calibrate a CTDM-5351 high definition camera from Sony, with a resolution of 752×582 pixels. To obtain a good accuracy, 20 pictures of the chessboard pattern have been taken in different positions and orientations. Figure 5.3 shows the toolbox graphical user interface (GUI) displaying the set of pictures. To perform an accurate calibration, the toolbox requires the number of squares along the horizontal and vertical directions, as well as the individual dimension of each square. In addition, for each chessboard image, one has to manually detect the four external corners of the chessboard. After repeating this process for all the n images, calibration is performed. As a practical example, the internal camera parameters matrix K for the CTDM-5351 high definition camera, as computed by the toolbox, is presented in (5.19):

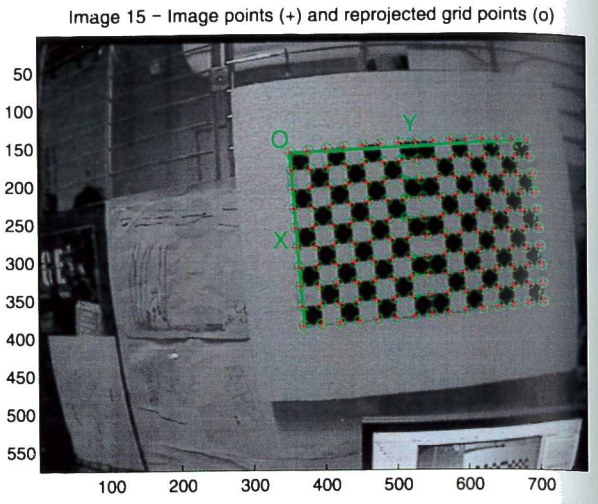


Fig. 5.4 Chessboard image photo with highlighted corners

$$K = \begin{bmatrix} 615.33303 & 0.0 & 377.68498 \\ & 614.07367 & 289.88436 \\ & & 1 \end{bmatrix} \tag{5.19}$$

Optionally, one can reproject on the chessboard images the corners used for calibration. Figure 5.4 shows a chessboard image photo with highlighted corners. From calibration, extrinsic parameters (rotation and translation) for each chessboard view are also computed. Figure 5.5(a) shows extrinsic parameters with coordinates centered in the camera, while Fig. 5.5(b) represent them in a world reference frame.

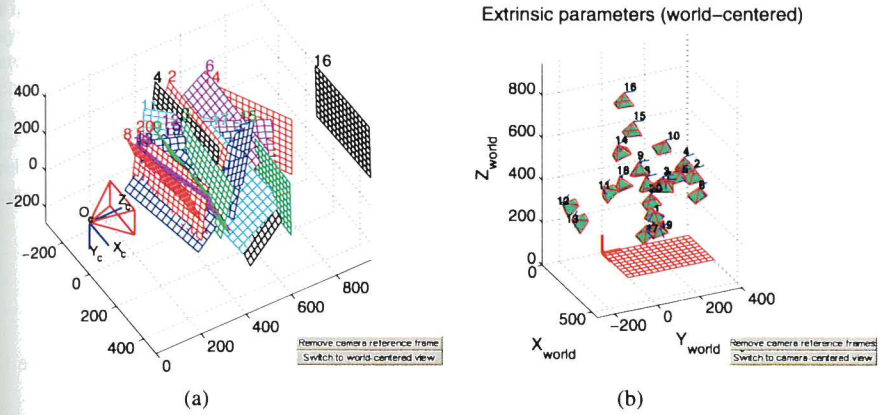
The camera calibration toolbox provides also a distortion model to compensate distortion caused by camera lenses. Figure 5.6 shows the resulting distortion model, considering radial and tangential distortion.

To demonstrate how distortion affects the camera's image, Fig. 5.7(a) shows an image as provided directly from camera, where straight lines appear curved. After applying the obtained model to correct radian and tangential distortion, the resulting image is shown in Fig. 5.7(b).

5.2 Stereo Imaging

Computers accomplish stereo imaging by finding correspondences between points that are seen by one imager and the same points as seen by the other imager. Then, the 3-dimensional location of the points can be estimated using such correspondences and a known baseline separation (geometry) between cameras, In practice, stereo imaging involves four steps:

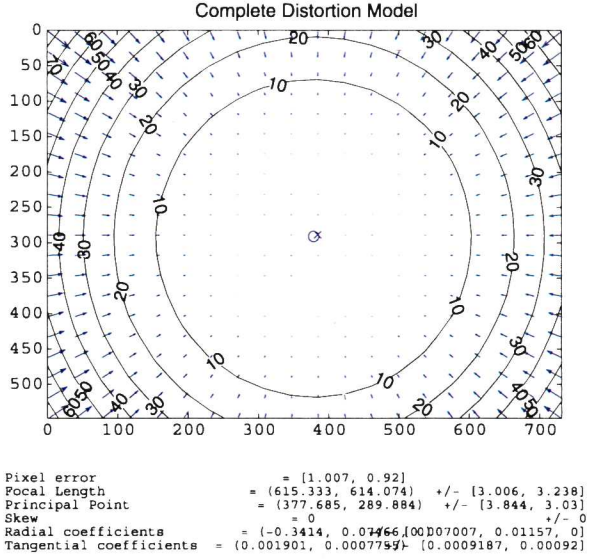
- *Undistortion*: mathematically remove radial and tangential lens distortion. Obtention of undistorted images.



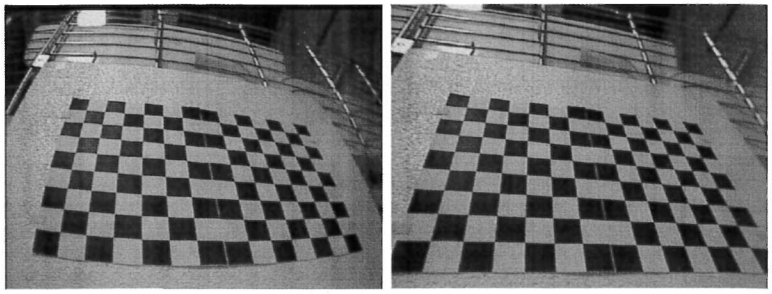
(a) (b)

Fig. 5.5 Extrinsic parameters: (**a**) coordinates centered in the camera; (**b**) world reference frame

Fig. 5.6 Distortion model for the CTDM-5351 camera



- *Rectification*: adjust for the angles and distances between cameras. Obtention of images that are row-aligned and rectified.
- *Correspondence*: finding the same features in the left and right camera views. Obtention of a disparity map, where the disparities are the differences in x-coordinates on the image planes of the $\mathbf{X}$ feature viewed in the left and right cameras: $\mathbf{x}_l - \mathbf{x}_r$.
- *Reprojection*: Knowing the geometric arrangement of the cameras, the disparity map can be turned into distances using *triangulation*. Obtention of a depth map.



(a) (b)

Fig. 5.7 Correcting distortion: (**a**) image as provided directly from camera; (**b**) corrected image

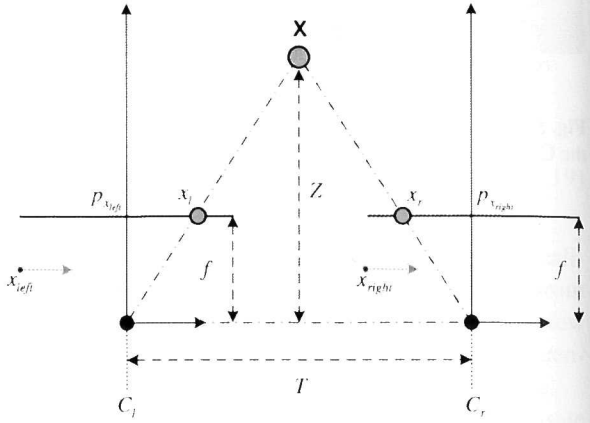


Fig. 5.8 Ideal stereo rig: The depth Z can be found by similar triangles. The principal rays of the imagers begin at the centers of projection C_l and C_r and extend through the principal points of the two image planes at $p_{x_{\text{left}}}$ and $p_{x_{\text{right}}}$

Triangulation Assume a perfectly undistorted, aligned, and measured stereo rig as shown in Fig. 5.8. Let us further assume a frontal parallel camera arrangement: the images are row-aligned and every pixel row of one camera aligns exactly with the corresponding row in the other camera. Consider one can find a point $\mathbf{X}$ in the physical world in the left and the right image views at $\mathbf{x}_l$ and $\mathbf{x}_r$, which will have the respective horizontal coordinates x_l and x_r. Taking x_l and x_r to be the horizontal positions of $\mathbf{x}_l$ and $\mathbf{x}_r$, respectively, allows to show that the depth is inversely proportional to the disparity between these views. The disparity is defined by $d = x_l - x_r$. From the schema shown in Fig. 5.8, one can easily derive the depth Z by using similar triangles

$$\frac{\mathbf{T} - (x_l - x_r)}{Z - f} = \frac{\mathbf{T}}{Z} \quad \Rightarrow \quad Z = \frac{f\mathbf{T}}{x_l - x_r} \tag{5.20}$$

There is obviously a nonlinear relationship between depth and disparity. When disparity is near 0, small disparity differences make for large depth differences. When disparity is large, small disparity differences do not change the depth by much. As a consequence, stereo vision systems have high depth resolution only for objects relatively near the camera.

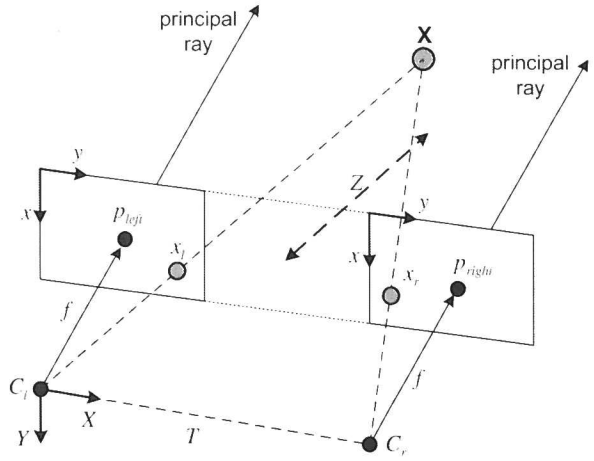


Fig. 5.9 Stereo coordinate system: the pixel coordinates are relative to the upper left corner of the image, and the two planes are row-aligned; the camera coordinates are relative to the left camera's center of projection

With the purpose of adapting the notation of this work to the notation used by the *Camera Calibration Toolbox for Matlab* [17] let us use a right-handed coordinate system as shown in Fig. 5.9. The left and right imager pixels have image origins at upper left in the image. Pixels are denoted by coordinates (x_l, y_l) and (x_r, y_r), respectively. The center of projection are at C_l and C_r; with principal rays intersecting the image plane at the principal point (p_x, p_y). After rectification, the cameras are row-aligned, displaced from one another by a distance $\mathbf{T}$, and with the same focal length f. With this arrangement it is relatively easily to solve for distance. In the real world, cameras will almost never be exactly aligned in the frontal parallel configuration, one must mathematically find image projections and distortion maps that will rectify the left and right images into such arrangement. When building a stereo rig, the cameras must be installed approximately frontal parallel and horizontally aligned, in order to make the mathematical transformations more tractable. If the cameras are not approximately aligned, the mathematical alignment can produce extreme image distortions, reducing the stereo overlap area of the resulting images.

The pair of cameras must capture their images at the exact same time, to avoid having problems if anything is moving in the scene, including the cameras themselves, therefore, the stereo cameras must be synchronized. Failing to do so, one limits oneself to using stereo vision with stationary cameras, viewing static scenes. Figure 5.10 shows the real situation between two cameras and the desired mathematical alignment. In order to perform this mathematical alignment, let us introduce some notations concerning the geometry of two cameras viewing a scene.

5.2.1 Epipolar Geometry

The basic geometry of a stereo imaging system is shown in Fig. 5.11. It is commonly referred to as *epipolar geometry*. For each camera there is a center of projection, C_l

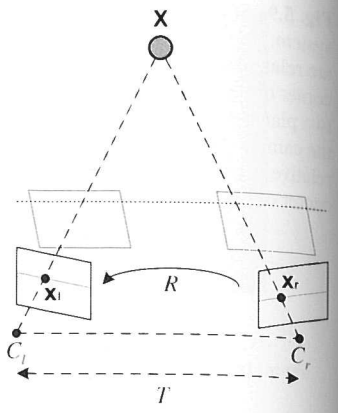


Fig. 5.10 The real situation between two cameras and the desired mathematical alignment

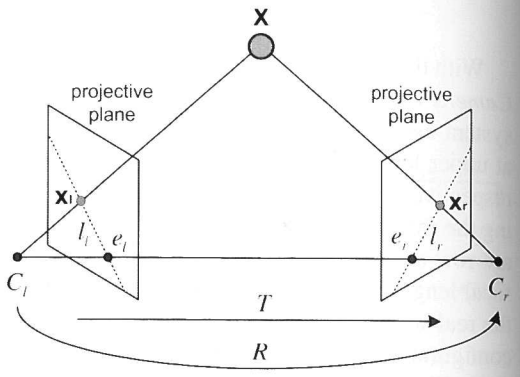


Fig. 5.11 Epipolar geometry: the basic geometry of a stereo imaging system

and C_r, and a pair of corresponding projective planes, P_l and P_r. The point $\mathbf{X}$ in the physical world has a projection onto each of the projective planes at $\mathbf{x}_l$ and $\mathbf{x}_r$, respectively. The new points of interest are the epipoles. An epipole e_l (respectively, e_r) on image plane P_l (respectively, P_r) is defined as the image of the center of projection of the other camera C_r (respectively, C_l). The plane in space formed by the viewed point $\mathbf{X}$ and the two epipoles e_l and e_r (or, equivalently, through the two centers of projection C_r and C_l) is called the epipolar plane, and the lines $\mathbf{x}_l e_l$ and $\mathbf{x}_r e_r$ (from the points of projection to the corresponding epipolar points) are called the epipolar lines.

Let us take for example the point $\mathbf{X}$ as seen by the camera on the right. Since that camera sees only $\mathbf{x}_r$ (the projection of $\mathbf{X}$ onto P_r), the actual point $\mathbf{x}_r$ could be located anywhere on the line defined by $\mathbf{x}_r$ and C_r. Such line, obviously, contains $\mathbf{X}$, as well as other points. Furthermore, its projection onto the left image plane P_l is the epipolar line defined by $\mathbf{x}_l$ and e_l. One concludes that the image of all of the possible locations of a point seen in one imager is the line that goes through the corresponding point and the epipolar point on the other imager. Let us summarize some facts about stereo camera epipolar geometry:

- Every 3-dimensional point in view is contained in an epipolar plane that intersects each image in an epipolar line.
- Given a feature in one image, its matching view in the other image must lie along the corresponding epipolar line (epipolar constraint).
- Once the epipolar geometry of the stereo rig is known, the 2-dimensional search for matching features across two imagers becomes a 1-dimensional search along the epipolar lines. This generates computational savings, and allows to reject a lot of points that could otherwise lead to spurious correspondences.
- Order is preserved. If points A and B are visible in both images and occur horizontally in that order in one imager, then they occur horizontally in that order in the other imager.

The Essential and Fundamental Matrices The essential matrix E contains information about the translation and rotation that relate the two cameras in physical space, see Fig. 5.11. E is purely geometrical and knows nothing about imagers. It relates the location, in physical coordinates, of the point $\mathbf{X}$ as seen by the left camera to the location of the same point as seen by the right camera (for which the notation $\mathbf{q}_l$ and $\mathbf{q}_r$ will be used).

The fundamental matrix F contains the same information as E, in addition to information about the intrinsics of both cameras. Therefore, F relates the points on the image plane of one camera in image coordinates (pixels) to the points on the image plane of the other camera in image coordinates (i.e., it relates $\mathbf{x}_l$ to $\mathbf{x}_r$).

Essential Matrix Math Given a point $\mathbf{X}$, one searches to derive a relationship between the observed locations $\mathbf{x}_l$ and $\mathbf{q}_r$ of $\mathbf{X}$ on the two imagers. This relationship is the essential matrix E. Let us use coordinates centered on C_l. In these coordinates, the location of the observed point is $\mathbf{X}_l$ and the origin of the other camera is located at $\mathbf{T}$. The point $\mathbf{X}$ as seen by the right camera is $\mathbf{X}_r$ in that camera's coordinates, where $\mathbf{X}_r = R[\mathbf{X}_l - \mathbf{T}]$. The key step is the introduction of the epipolar plane, which relates all of these things. Let us recall that the equation for all points $\mathbf{x}$ on a plane with normal vector $\mathbf{n}$ and passing through point $\mathbf{a}$ obeys the constraint $(\mathbf{x}-\mathbf{a})\cdot\mathbf{n} = 0$.

The epipolar plane contains the vectors $\mathbf{X}_l$ and $\mathbf{T}$, thus, a vector $\mathbf{X}_l \times \mathbf{T}$ perpendicular to both can be used instead of $\mathbf{n}$ in the plane equation. Therefore, en equation for all possible points $\mathbf{X}_l$ through the point $\mathbf{T}$ and containing both vectors would be $[\mathbf{X}_l - \mathbf{T}]^{\mathrm{T}}[\mathbf{T} \times \mathbf{X}_l] = 0$. The objective is to relate $\mathbf{x}_l$ and $\mathbf{x}_r$ by first relating $\mathbf{X}_l$ and $\mathbf{X}_r$. Let us draw $\mathbf{X}_r$ into the picture via the equality $\mathbf{X}_r = R[\mathbf{X}_l - \mathbf{T}]$, which can be rewritten as $[\mathbf{X}_l - \mathbf{T}] = R^{-1}\mathbf{X}_r$. Making this substitution and given that $R^{\mathrm{T}} = R^{-1}$ yields $[R^{\mathrm{T}}\mathbf{X}_r]^{\mathrm{T}}[\mathbf{T} \times \mathbf{X}_l] = 0$. Since it is possible to rewrite a cross product as a matrix multiplication, let the matrix S be such that

$$\mathbf{T} \times \mathbf{X}_l = S\mathbf{X}_l = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \mathbf{X}_l \tag{5.21}$$

Making this substitution for the cross product gives $\mathbf{X}_r^{\mathrm{T}} RS\mathbf{X}_l = 0$. The essential matrix E is defined by the product RS, leading to the compact equation $\mathbf{X}_r^{\mathrm{T}} E\mathbf{X}_l = 0$. In

order to obtain the relation between the points as one observes them on the images, let us substitute using the projection equations

$$\mathbf{q}_l = f_l \frac{\mathbf{X}_l}{Z_l}; \qquad \mathbf{q}_r = f_r \frac{\mathbf{X}_r}{Z_r}$$

and then divide them by $\frac{Z_l Z_r}{f_l f_r}$ to obtain the final result:

$$\mathbf{q}_r^{\mathrm{T}} E \mathbf{q}_l = 0 \tag{5.22}$$

The 3×3 essential matrix E has rank 2 (rank-deficient matrix), which actually ends up being an equation for a line. There are five parameters in E: three for rotation and two for the direction of translation (scale is not set), along with two additional constraints:

- The determinant is 0 because it is rank-deficient.
- Its two nonzero singular values are equal because the matrix S is skew-symmetric and R is a rotation matrix.

Yielding a total of seven constraints.

Fundamental Matrix Math In order to find a relationship between a pixel in one image and the corresponding epipolar line in the other image, one has to introduce intrinsic information about the two cameras. To do this, for $\mathbf{x}$ (the pixel coordinate) one substitutes $\mathbf{q}$ and the camera intrinsics matrix that relates them. Recall that $\mathbf{x} = K\mathbf{q}$ (where K is the camera intrinsics matrix) or, equivalently, $\mathbf{q} = K^{-1}\mathbf{x}$. Hence the equation for E becomes

$$\mathbf{x}_r^{\mathrm{T}} K_r^{-\mathrm{T}} E K_l^{-1} \mathbf{x}_l = 0$$

by defining the fundamental matrix F as $K_r^{-\mathrm{T}} E K_l^{-1}$ now one can write

$$\mathbf{x}_r^{\mathrm{T}} F \mathbf{x}_l = 0 \tag{5.23}$$

Whereas E operates in physical coordinates, the fundamental matrix F operates in image pixel coordinates. F is of rank 2 and has seven parameters, two for each epipole and three for the homography that relates the two image planes (the scale aspect is missing from the usual four parameters).

Epipolar Lines Equation (5.23) is true, because if points $\mathbf{x}_l$ and $\mathbf{x}_r$ correspond, then $\mathbf{x}_r$ lies on the epipolar line $l_r = F\mathbf{x}_l$ corresponding to the point $\mathbf{x}_l$. In other words $0 = \mathbf{x}_r^{\mathrm{T}} l_r = \mathbf{x}_r^{\mathrm{T}} F \mathbf{x}_l$. Conversely, if image points satisfy the relation $\mathbf{x}_r^{\mathrm{T}} F \mathbf{x}_l = 0$ then the rays defined by these points are coplanar. This is a necessary condition for points to correspond.

5.2.2 *Calibration of the Stereo Imaging System*

Stereo calibration is the process of computing the geometrical relationship between the two cameras in space. It depends on finding the rotation matrix R and translation

vector $\mathbf{T}$ that relate the right camera to the left camera, as depicted in Fig. 5.11. For any given 3-dimensional point $\mathbf{X}$, one can separately use a single-camera calibration to put $\mathbf{X}$ in the camera coordinates $\mathbf{X}_l = R_l \mathbf{X} + \mathbf{T}_l$ and $\mathbf{X}_r = R_r \mathbf{X} + \mathbf{T}_r$ for the left and right cameras, respectively. Note in Fig. 5.11 that both views of $\mathbf{X}$ are related by $\mathbf{X}_l = R^{\mathrm{T}}(\mathbf{X}_r - \mathbf{T})$, where R and $\mathbf{T}$ are, respectively, the rotation matrix and translation vector between the cameras. Taking these three equations and solving for the rotation and translation separately yields the following simple relations:

$$R = R_r R_l^{\mathrm{T}} \tag{5.24}$$

$$\mathbf{T} = \mathbf{T}_r - R\mathbf{T}_l \tag{5.25}$$

Given a set of paired views of chessboard corners, the *Camera Calibration Toolbox for Matlab* solves for rotation and translation parameters of the chessboard views, for each camera separately. It then plugs these left and right rotation and translation solutions into (5.24) and (5.25) to solve for the rotation and translation parameters between the two cameras. Image noise and rounding errors cause that each chessboard pair results in slightly different values for R and $\mathbf{T}$. The calibration routine takes the median values for R and $\mathbf{T}$ as an initial approximation of the true solution, then, it runs a robust Levenberg–Marquardt iterative algorithm to find the minimum of the reprojection error of the chessboard corners for both camera views, and the solution for R and $\mathbf{T}$ is returned.

Stereo calibration gives the rotation matrix that will put the right camera in the same plane as the left camera, this makes the two image planes coplanar but not row-aligned; to do so, a stereo rectification must be accomplished.

Stereo Rectification It is easiest to compute the stereo disparity when the two image planes align exactly. Unfortunately, when using a real stereo system, a perfectly aligned configuration is rare since the two cameras almost never have exactly coplanar, row-aligned imaging planes. Figure 5.10 shows the goal of stereo rectification. One seeks to reproject the image planes of the two cameras so that they reside in the exact same plane, with image rows perfectly aligned into a frontal parallel configuration.

One wants the image rows between the two cameras to be aligned after rectification, so that stereo correspondence will be more reliable and computationally tractable. By having to search only one row for a match with a point in the other image, reliability and computational efficiency are both enhanced. The result of aligning horizontal rows within a common image plane containing each image is that the epipoles themselves are then located at infinity. That is, the image of the center of projection in one image is parallel to the other image plane. Since there are an infinite number of possible frontal parallel planes to choose from, one needs to add more constraints, like maximizing view overlap and minimizing distortion.

Eight terms will result from the process of aligning the two image planes. For each camera one obtains a distortion vector, a rotation matrix, and the rectified and unrectified camera matrices (K_{rect} and K, respectively). From these terms, a map for interpolating pixels from the original image can be constructed, in order to create a new rectified image. To compute the rectification terms, the *Camera Calibration*

Toolbox for Matlab implements Bouguet's algorithm, which uses the rotation and translation parameters from two calibrated cameras viewing a calibration pattern.

Calibrated Stereo Rectification: Bouguet's Algorithm Given R and $\mathbf{T}$ between the stereo images, Bouguet's algorithm attempts to minimize the amount of change reprojection produces for each of the two images while maximizing common viewing area. To minimize image reprojection distortion, the rotation matrix R that rotates the right camera's image plane into the left camera's image plane is split in half between the two cameras, resulting r_l and r_r rotation matrices for the left and right camera, respectively.

Each camera rotates half a rotation, so their principal rays each end up parallel to the vector sum of where their original principal rays had been pointing, putting the cameras into coplanar alignment but not into row alignment. To compute the R_{rect} that will take the left camera's epipole to infinity and align the epipolar lines horizontally, one creates a rotation matrix by starting with the direction of the epipole $\mathbf{e}_l$ itself. Taking the principal point (p_x, p_y) as the left image's origin, the direction of the epipole is directly along the translation vector between the two camera's centers of projection. The next vector $\mathbf{e}_2$, must be orthogonal to $\mathbf{e}_1$ but is otherwise unconstrained. For $\mathbf{e}_2$, choosing a direction orthogonal to the principal ray is a good choice. This is accomplished by using the cross product of $\mathbf{e}_1$ with the direction of the principal ray and then normalizing, so that obtaining another unit vector. The $\mathbf{e}_3$ is just orthogonal to $\mathbf{e}_1$ and $\mathbf{e}_2$, it can be found using their cross product. Then, $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$ are expressed as

$$\mathbf{e}_1 = \frac{T}{\|T\|}; \qquad \mathbf{e}_2 = \frac{[-T_y \; T_x \; 0]^{\mathrm{T}}}{\sqrt{T_x^2 + T_y^2}}; \qquad \mathbf{e}_3 = \mathbf{e}_1 \times \mathbf{e}_2$$

The matrix that takes the epipole in the left camera to infinity is then

$$R_{\text{rect}} = \begin{bmatrix} \mathbf{e}_1^{\mathrm{T}} \\ \mathbf{e}_2^{\mathrm{T}} \\ \mathbf{e}_3^{\mathrm{T}} \end{bmatrix}$$

which rotates the left camera about the center of projection so that the epipolar lines become horizontal and the epipoles are at infinity. The row alignment of the two cameras is achieved by setting $R_l = R_{\text{rect}} r_l$ and $R_r = R_{\text{rect}} r_r$. The rectified left and right camera matrices $K_{l_{\text{rect}}}$ and $K_{r_{\text{rect}}}$ are also computed, but returned combined with projection matrices Π_l and Π_r:

$$\Pi_l = K_{l_{\text{rect}}} \Pi_l' = \begin{bmatrix} \alpha_{x_l} & s_l & x_{0_l} \\ 0 & \alpha_{y_l} & y_{0_l} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.26}$$

$$\Pi_r = K_{r_{\text{rect}}} \Pi_r' = \begin{bmatrix} \alpha_{x_r} & s_r & x_{0_r} \\ 0 & \alpha_{y_r} & y_{0_r} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.27}$$

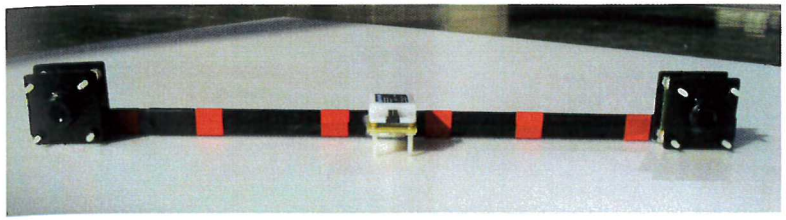


Fig. 5.12 The stereo rig: uEye UI-1226LE-M-G cameras installed with a separation of 35 cm

The projection matrices take a 3-dimensional point in homogeneous coordinates to a 2-dimensional point in homogeneous coordinates as

$$\Pi \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ w \end{bmatrix} \tag{5.28}$$

Screen coordinates can be calculated as $(x/w, y/w)$. 2-dimensional points can also then be reprojected into three dimensions given their screen coordinates and the camera intrinsics matrix. The reprojection matrix is defined as

$$Q = \begin{bmatrix} 1 & 0 & 0 & -x_{0_l} \\ 0 & 1 & 0 & -y_{0_l} \\ 0 & 0 & 0 & f \\ 0 & 0 & -\frac{1}{T_x} & \frac{(x_{0_l} - x_{0_r})}{T_x} \end{bmatrix} \tag{5.29}$$

If the principal rays intersect at infinity, then $x_{0_l} = x_{0_r}$ inf and the term in the lower right corner is 0. Given a 2-dimensional homogeneous point and its associated disparity d, the point can be projected into three dimensions by

$$Q \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} \tag{5.30}$$

The 3-dimensional coordinates are then $(X/W, Y/W, Z/W)$. Bouguet's rectification method yields the ideal stereo configuration shown in Fig. 5.8. New image centers and new image bounds are then chosen for the rotated images so as to maximize the overlapping viewing area.

Calibrating a Stereo Rig Using the *Camera Calibration Toolbox for Matlab* A stereo rig has been constructed with the purpose of performing stereo imaging studies. Two *uEye UI-1226LE-M-G* cameras from *IDS GmbH* [48] have been installed with a separation of 35 cm between them, as shown in Fig. 5.12. Bouguet's stereo calibration method *Camera Calibration Toolbox for Matlab* requires first that both cameras are individually calibrated. For this purpose, the method previously presented in Sect. 5.1.1 has been used to obtain the calibration parameters of the

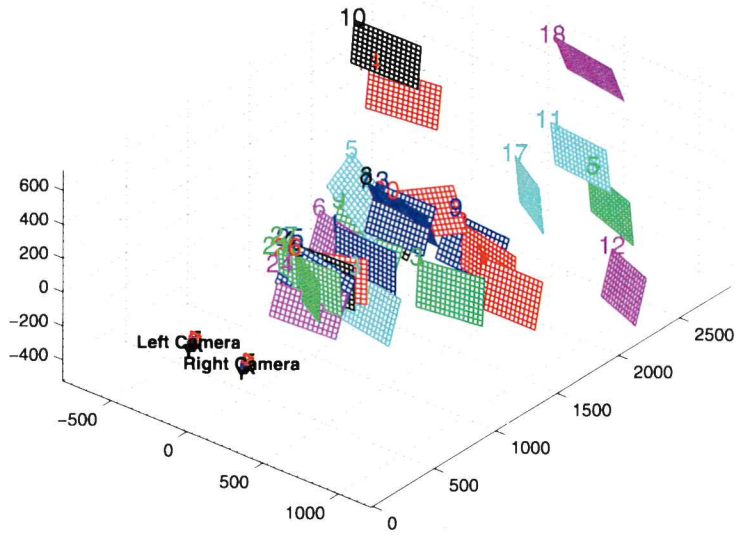


Fig. 5.13 The stereo rig and the chessboard positions

pair of uEye cameras. Using those parameters, the toolbox estimates the position of the chessboard pattern with respect to the stereo rig, and generates an image to illustrate them, it can be seen in Fig. 5.13. Once calibration is performed, the toolbox provides the next data,

$$K_l = \begin{bmatrix} 369.30485 & 0.0 & 173.42147 \\ 0.0 & 372.40078 & 139.26367 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{5.31}$$

$$K_r = \begin{bmatrix} 368.41268 & 0.0 & 200.22282 \\ 0.0 & 369.24829 & 138.03250 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{5.32}$$

$$R = \begin{bmatrix} 0.9969 & 0.0057 & -0.0778 \\ -0.0072 & 0.9998 & -0.0185 \\ 0.07775 & 0.0191 & 0.9967 \end{bmatrix} \tag{5.33}$$

$$\mathbf{T} = \begin{bmatrix} -354.9561 & 9.3886 & -19.6539 \end{bmatrix}^{\mathrm{T}} \tag{5.34}$$

where K_l and K_r represent the left and right camera matrices, respectively, R is the rotation matrix that rotates the right camera's image plane into the left camera's image plane, and $\mathbf{T}$ is the translation vector that relates the right camera with the left camera. The essential matrix is then described by

$$E = \begin{bmatrix} 0.6175 & 47.2370 & 11.4110 \\ -19.4754 & 6.4582 & 354.8174 \\ -9.7339 & -352.2883 & 7.5109 \end{bmatrix} \tag{5.35}$$

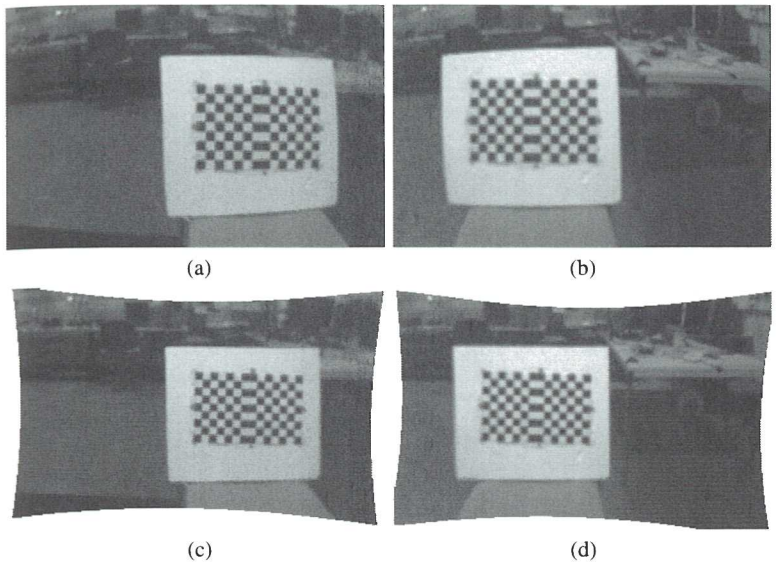


Fig. 5.14 Stereo pairs: (**a**) left image; (**b**) right image; (**c**) rectified left image; (**d**) rectified right image

and the fundamental matrix is

$$F = \begin{bmatrix} 0.000004 & 0.000344 & -0.017762 \\ -0.000142 & 0.000046 & 0.979145 \\ -0.007552 & -1.021412 & 12.22679 \end{bmatrix} \tag{5.36}$$

The obtained projection matrices for the left and right cameras are then

$$\Pi_l = \begin{bmatrix} 369.30485 & 0.0 & 173.42147 & 0.0 \\ 0.0 & 372.40078 & 139.26367 & 0.0 \\ 0.0 & 0.0 & 1.00000 & 0.0 \end{bmatrix} \tag{5.37}$$

$$\Pi_r = \begin{bmatrix} 382.855 & 5.9537 & 170.8889 & -134705.5132 \\ 8.0565 & 371.8116 & 130.7228 & 753.8591 \\ 0.0777 & 0.0191 & 0.9967 & -19.6539 \end{bmatrix} \tag{5.38}$$

Figure 5.14(a)–(b) shows one of the left and right images pairs used during the stereo calibration process. Applying the calibration parameters obtained allows rectifying the image pair, the result is shown in Fig. 5.14(c)–(d). It can be verified that after rectification, the overlapping viewing area has been maximized. Some 3-dimensional reconstructions have been performed with the purpose of validating the accuracy of the stereo calibration. The four corners of the white square surrounding the chessboard have been selected as the desired features to reconstruct. From the upper left corner, and in a clock-wise rotation, the four corners are located in the left and right images, respectively, at

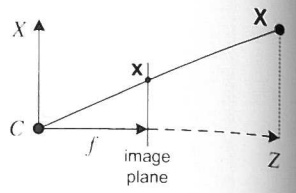


Fig. 5.15 Projective geometry for optical flow computing

$$\mathbf{C}_{1_{\text{left}}} = \begin{bmatrix} 153 \\ 49 \end{bmatrix}; \quad \mathbf{C}_{1_{\text{right}}} = \begin{bmatrix} 49 \\ 51 \end{bmatrix}$$

$$\mathbf{C}_{2_{\text{left}}} = \begin{bmatrix} 324 \\ 53 \end{bmatrix}; \quad \mathbf{C}_{2_{\text{right}}} = \begin{bmatrix} 221 \\ 41 \end{bmatrix}$$

$$\mathbf{C}_{3_{\text{left}}} = \begin{bmatrix} 327 \\ 197 \end{bmatrix}; \quad \mathbf{C}_{3_{\text{right}}} = \begin{bmatrix} 224 \\ 194 \end{bmatrix}$$

$$\mathbf{C}_{4_{\text{left}}} = \begin{bmatrix} 159 \\ 208 \end{bmatrix}; \quad \mathbf{C}_{4_{\text{right}}} = \begin{bmatrix} 54 \\ 197 \end{bmatrix}$$

The 3-dimensional reconstruction of each corner, respectively, result is

$$\mathbf{C}_1 = \begin{bmatrix} -0.2496 \\ -0.3113 \\ 1.3552 \end{bmatrix}$$

$$\mathbf{C}_2 = \begin{bmatrix} 0.3586 \\ -0.3146 \\ 1.3093 \end{bmatrix}$$

$$\mathbf{C}_3 = \begin{bmatrix} 0.3650 \\ 0.2083 \\ 1.2980 \end{bmatrix}$$

$$\mathbf{C}_3 = \begin{bmatrix} -0.2267 \\ 0.2305 \\ 1.3163 \end{bmatrix}$$

with values expressed in meters. The 3-dimensional reconstruction is performed with satisfactory results, since the real position of the chessboard during the experiment was 1.3 m in front of the cameras. Note also how the (x, y) estimations of each corner are consistent with their corresponding 3-dimensional position.

Using the mathematical relationships just derived, the stereo vision system can be installed onboard an autonomous agent with the purpose of estimating its 3-dimensional position with respect to its surrounding environment.

5.3 Optical Flow

Optical flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer (the imaging sensor), and the scene. The optical flow is created by the translational and the rotational movements of a point $\mathbf{X} = (X, Y, Z)$ on the camera reference frame. Consider the projection of $\mathbf{X}$ into the image plane, $\mathbf{x} = (x, y)$, as shown in Fig. 5.15, its time derivative in function of the point $\mathbf{X}$ can be expressed as

$$\frac{d\mathbf{x}}{dt} = \frac{1}{Z}\frac{d\mathbf{X}}{dt} \tag{5.39}$$

with

$$\frac{d\mathbf{X}}{dt} = \mathbf{V} + R \times \mathbf{X} \tag{5.40}$$

where $\mathbf{V}$ represents the translational movement and R the rotational movement of the point. Then, optical flow can be expressed as

$$\begin{bmatrix} \text{OF}_x \\ \text{OF}_y \end{bmatrix} = \mathbf{T}_{\text{OF}} + \mathbf{R}_{\text{OF}} \tag{5.41}$$

with the translational part

$$\mathbf{T}_{\text{OF}} = \frac{1}{Z}\begin{bmatrix} -f & 0 & x \\ 0 & -f & y \end{bmatrix}\begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} \tag{5.42}$$

and the rotational part

$$\mathbf{R}_{\text{OF}} = \begin{bmatrix} \frac{xy}{f} & -(f + \frac{x^2}{f}) & y \\ (f + \frac{y^2}{f}) & -\frac{xy}{f} & -x \end{bmatrix}\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{5.43}$$

where OF_* is the optical flow component in the (x or y) coordinate of the point $\mathbf{x}$, V_* and ω_* are the translational and rotation velocities rates, respectively, of the point $\mathbf{X}$. Equation (5.41) represents the optical flow defined on the image plane. However, optical flow can also be defined over other projection surfaces, such as a sphere, which is often used for its passivity properties. If we apply the appropriate mathematical relationships when using a different projection surface, optical flow information is accurately estimated.

5.3.1 Computing Methods

The standard techniques for calculating the optical flow can be classified in four main groups: differential or gradient methods based on intensity [45, 58], methods of correlation or block matching [7, 75], methods based on energy [42] and those based on phase [36, 80]. Block matching techniques present good accuracy and performance against aperture problems and large displacements, unfortunately they are computationally expensive, their accuracy decreases in the presence of deformation, and displacements of less than one pixel are not detectable. Energy-based and phase-based methods are computationally expensive and their implementation

is very complicated. Due to the derivatives estimation, differential methods present high sensitivity issues caused by poor lighting or image noise. However, they have good precision and their processing is less time consuming than all the other cited techniques. From all the discussed methods, the differential approach is the most implemented. Indeed, differential methods are well known and widely used in the literature for the calculation of optical flow.

The optical flow differential method is based on the computation of the spatial-temporal derivatives of image intensity over a large image region (global method) or local neighborhoods (local methods). Let $I(x,t)$ be a 1-dimensional image, and suppose the intensity remains the same, except if it shifts right or left at constant velocity V_x. Therefore, the spatial derivative $I_x = \frac{\partial I}{\partial x}$ and the temporal derivative $I_t = \frac{\partial I}{\partial t}$ follow the rule

$$I_t = -V_x I_x \tag{5.44}$$

For 2-dimensional images, assuming that the intensity $I(x,y,t)$ structure is constant in a local time-varying region, one writes

$$I(x,y,t) = I(x+\delta x, y+\delta y, t+\delta t) \tag{5.45}$$

where t is the time and $(\delta x, \delta y)$ is the image displacement after time δt. From (5.45) it is possible to obtain the constraint of conservation of intensity, expressed by

$$\nabla I(x,y)(v_x, v_y) + I_t = 0 \tag{5.46}$$

In [58] the authors have constructed a technique for estimating the optical flow, which is known as the Lucas–Kanade algorithm. It is based on a weighted least squares minimization of the intensity conservation constraint in each small spatial neighborhood

$$\min \sum W^2(\mathbf{x}) \left[\nabla I(x,y)v + I_t(\mathbf{x},t)\right]^2 \tag{5.47}$$

where W is a window that gives more importance to the constraints near the center of the chosen neighborhood. The solution of (5.47) is given by

$$v = \left[\mathbf{A}^{\mathrm{T}} W^2 \mathbf{A}\right]^{-1} \mathbf{A}^{\mathrm{T}} W \mathbf{b} \tag{5.48}$$

where

$$\mathbf{A} = \left[\nabla I(\mathbf{x}_i), \ldots, I(\mathbf{x}_n)\right]^{\mathrm{T}} \tag{5.49}$$

$$W = \mathrm{diag}\left[W(\mathbf{x}_i), \ldots, I(\mathbf{x}_n)\right] \tag{5.50}$$

$$\mathbf{b} = -\left[I_t(\mathbf{x}_1), \ldots, I_t(\mathbf{x}_n)\right]^{\mathrm{T}} \tag{5.51}$$

An interesting characteristic of the Lucas–Kanade algorithm is that it provides a measure of the estimation error, given that the matrix $[\mathbf{A}^{\mathrm{T}} W^2 \mathbf{A}]^{-1}$ is consistent with a covariance matrix. Therefore, unreliable estimates can be identified using the inverse eigenvalues of this matrix. However, this method is not suitable for dealing with displacements exceeding a pixel per frame, causing the estimation to fail.

Nevertheless, an extension of this method that computes the optical flow via a hierarchical coarse-to-fine process has been proposed. It is called the Lucas–Kanade pyramidal representation, and complements the original method with a pseudo-iterative scheme, allowing to compute the optical flow by propagating the flow in lower resolutions to larger resolutions [16].

In spite of all the advantages of the Lucas–Kanade algorithm, the optical flow estimated by this method represents the apparent movement of the objects in the scene, which may not correspond to the real object movement. In order to compute effectively the optical flow, the next points should be taken into account:

- The lightning has to stay constant over time. This hypothesis is the basis of the optical flow computation. Without it, (5.45) could not be written. If the illumination of the scene changes from one instant to another, objects may seem like being on movement, when in reality they are static over time. Indeed, according to the position of the light source, shadows are different, and even if the object is stationary, the imaging sensor can detect the movement of shadows.
- A rich-textured image is another important condition for accurate estimation of the optical flow. For a differential approach, like the Lucas–Kanade algorithm, the most contrasting textures have to be chosen in order to perform properly. A low-pass filter can be applied with the purpose of incrementing the image's contrast. Commonly, a Gaussian filter is used to create a sort of blur to the image. However, a simpler way of achieving the same effect consists of defocusing the lens, so the image becomes blurred. Nevertheless, filtering has to be performed carefully, since computing the optical flow requires good contrasts, but they should not be too discontinuous.
- A differential method alone does not solve what is called the *aperture problem*, which arises when using a small window to measure motion: one often sees only an edge, not a corner. But an edge alone is insufficient to determine exactly how (i.e., in what direction) the entire object is moving. In conclusion, one should observe more globally. However, this implies incrementing the computing time. A solution to this problem consists of eliminating the inconsistent points by means of the covariance matrix previously introduced.

Optical flow algorithms, like the Lucas–Kanade algorithm, are designed to work well with conventional CCD cameras. In the present studies, a conventional imaging sensor in combination with the pyramidal implementation of the Lucas–Kanade algorithm is used for computing optical flow.

5.4 Implementing an Imaging System for the Quad-Rotor UAV

Nowadays technological advances allow the development of vision systems well suited for being implemented on mini-UAVs. If imaging processing is intensive, the onboard imaging sensor can be connected to a wireless transmitter. This allows sending real-time video signal to a remote computer equipped with a video receiver.

Once images are received, a computer vision application processes the real-time video and extracts the required information. With the purpose of generating remote control commands for stabilizing the vehicle in flight, the imaging-based data can be used by a supervisory control program, running also in the remote computer. In a different approach, image-based measurements can be sent back directly to the aerial vehicle, in order to perform a data fusion and a control strategy onboard.

Recently, the robotics research community has increased its interest in the development of fully embedded video-processing systems for UAVs. However, the development of such system, for a mini-UAV like the quad-rotor, represents a very challenging task. Imaging processing is computationally expensive, therefore, it must be performed in a powerful processor. Unfortunately, the quad-rotor payload capacity is very limited, restricting considerably the available options for embedded computers. This situation motivated several research groups to develop their own prototypes and solutions for performing image processing.

This section presents first a brief discussion concerning deported and embedded vision systems. The following subject addresses several important characteristics that must be considering when implementing a monocular or a stereo imaging system onboard a quad-rotor UAV. Next, the development of a monocular and a stereo vision system for the quad-rotor platform are detailed. Both systems have been built with the purpose of evaluating the benefits and drawbacks of each approach in real-time experiments.

5.4.1 Deported and Embedded Systems

Nowadays, technological advances allows performing image processing onboard the flying robot or in a deported ground station PC. Generally, the deported solution is chosen when the computational tasks requires more resources than one could embed in the vehicle. Some vision algorithms such as optical flow estimation and SLAM schemes require working with dense fields (great number of points) and iterative approaches which are computational expensive. When imaging tasks are computationally expensive, the best solution is to use fast top-technology computers. However, these last ones are commonly heavy and consume too much power, which are two contradictory characteristics with respect to the design constraints of unmanned aerial vehicles. Nevertheless, the UAV research domain is taking advantage of the efforts that the electronic industry is doing due to the demand of developing small and powerful devices such as mobile phones, which are becoming smart devices. Thus, it is sure that, in a short time, all kinds of visual processing will be done onboard with very powerful, tiny and low power microcomputers. In the following sections, the different solutions associated with embedded and deported systems will be detailed, taking in consideration the quad-rotor design constraints previously explained.

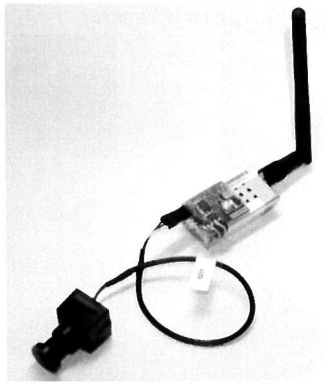


Fig. 5.16 Analog camera and video transmitter set-up

5.4.1.1 Deported Systems

Deported systems are based on wireless technology. Nowadays, there exist two different ways of realize this wireless connection: *wifi* and *radio frequency*. Radio frequency imaging system are basically conformed of an analog camera connected directly to a video transmitter. Figure 5.16 shows an analog camera–video transmitter set-up. This device sends the camera image through radio frequency data to the receiver placed on the ground, which in turns transfers the analog image to a frame grabber that performs the data conversion. Finally, the digitalized image is provided to the computer via USB. The communication between the video transmitter and receiver is executed at standard pre-defined working frequencies. There exist three standard frequencies allowed for video transmitting: 2.45 GHz, 1.2 GHz and 5.8 GHz. The most common frequency band used is the 2.45 GHz band, which is often very saturated by other wireless communications in certain areas. Generally, it is possible to choose compatible channels between the different communications present in the band. However, when a pair of devices are not built with the same frequency standard, it is impossible to suppress the interferences that could exist between them.

The principal disadvantage of radio frequency transmission is that it presents unwanted issues related to multipath signal transfer, which occur mostly in indoor environments where walls, furniture and other objects interfere with the signal's path occasioning signal rebounds. Multipath problems can cause blur and noisy images, which are very difficult to work with. For this reason, special attention is required in the way that the overall system handles noisy images since a control input resulting from a false image measurement can instantaneously cause the crash of the vehicle.

Visual systems working with wifi use a digital IP camera that establishes a TCP/IP network between the ground computer and the camera itself, see Fig. 5.17. The camera's image is transferred compressed in a digital format. Then, the ground computer has to decompress the image in order to be able of continuing with further processing. Generally the decompressing stage could take a considerably amount of time, causing a delay in the overall process of up to 1 ms. This situation results in

Fig. 5.17 Digital IP camera

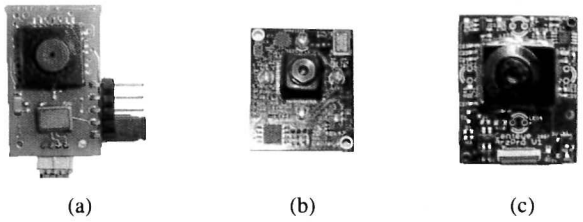
a considerably lag between the instant of time when the image is taken and the moment when the respective control input arrives at the UAV. Obviously this situation is delicate when the stability of the aerial robot and the performance of the imaging system is executed in real-time. Nevertheless, the principal advantage of this digital transmission is that the data are not perturbed by other signals, which results in clear and noiseless images at the reception device. It is also worth to mention that deported systems are very useful when the weight constraint of the UAV is in its limits and when the computation process is very expensive.

5.4.1.2 Embedded Systems

Embedded visual system not only contains the optical sensor, but also has the computing resources to process the captured images. The principal advantage of such systems is that no lag is introduced in the imaging processing loop. However, depending on the capabilities of the onboard system processor, the working frequency could be more or less affected compared to the one working frequencies reached by an ordinary computer. Indeed, if the processor is not powerful enough to running the algorithm, the performance of the embedded systems will be worse that the one of the deported systems. Until recent years, the only embedded systems that could be carried by the quad-rotor were some dedicated optical flow sensors and the CMUCam. For example, the VLSI sensor developed presented in [10] was patented and first commercialized under the name of LadyBug sensor, see Fig. 5.18(a), then it was upgraded to the Mantis sensor, see Fig. 5.18(b), and finally to the Arz sensor Fig. 5.18(c).

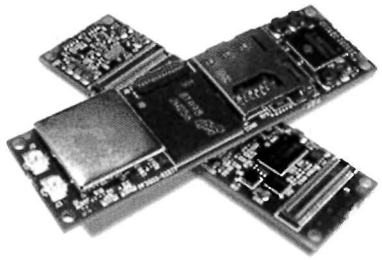
The CMUCam3 is another interesting visual embedded system first developed by the Carnegie Mellon University. Such system is programmable, allowing the users to adapt their own algorithms in order to be executed by the processor available on the system. However, this is a low-cost solution where most of the complex algorithms

Fig. 5.18 Centeye optical flow sensors: (**a**) LadyBug; (**b**) Mantis; (**c**) Arz



(a) (b) (c)

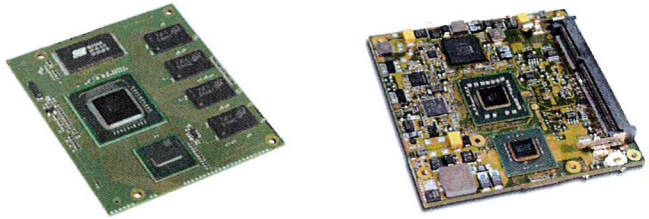
Fig. 5.19 Gumstix Overo Fire



already adapted for this system run a low rates, which is not appropriate for aerial vision-based navigation.

Nowadays, with the size and weight reductions of high-performance processors, computer-on-module systems are taking an important place in the market. The computer vision field is taking advantage of those developments to finally implement very complex and accurate algorithms in small size devices. The Gumstix Overo Fire, see Fig. 5.19, belongs to the family of computer-on-module systems. With a tiny ARM Cortex-A8 OMAP3530 processor, a TI C64x DSP and the POWERVR SGX for graphics acceleration, the Overo Fire can handle time-consuming algorithms that require considerably high computing resources. Different teams working on UAVs have started to developed visual systems based on this device with very promising results. However, this device still has a limited range of applications.

In addition to those systems, Ascending Technologies have designed a computer-on-module specially adapted for visual applications on UAVs. The system is conformed by a CoreExpress board from Lippert Embedded, see Fig. 5.20(a). Since the processor is a X86 architecture, standard operating systems and drivers for devices like cameras, laser range finders or WiFi could be used. The development of such powerful computer-on-module has attracted the attention to other systems even more powerful. The PIXHAWK team from the Computer Vision and Geometry Lab at the ETH Zurich have developed a computer-on-module baseboard to carry the microETXexpress-PC from Kontron, see Fig. 5.20(b). This system is a very powerful computer-on-module, consisting of an Intel Core 2 Duo SL9400 running at 1.86 GHz with 6 MB L2 Cache, a 2 GB DDR3 memory chipset, and the Intel GMA X4500 integrated graphics. This is a very complete system that can be compared to a netbook computer. However, it weighs 235 grams, which could be a serious problem when carried by a quad-rotor vehicle. The implementation of such system



(a) (b)

Fig. 5.20 Computer-on-module sensors: (**a**) Lippert Embedded CoreExpress board; (**b**) Kontron microETXexpress-PC

on the overall design of the aerial vehicle requires the optimization of the rest of the vehicle, in order to save weight for the vision computer.

5.4.2 *Challenges when Using Monocular and Stereo Imaging Systems*

With the purpose of estimating relative positioning and velocity where other sensing technologies cannot, monocular and stereo vision systems are being installed onboard UAVs. A discussion concerning important characteristics of each one of those systems is given, as well as some challenges, advantages and disadvantages, encountered during their implementation.

Monocular Imaging System From a set of consecutive images, motion as well as 3-dimensional structure can be estimated by a single camera. First of all, the essential matrix E or fundamental matrix F describing the relative motion between two views has to be estimated. This can be achieved by using feature correspondences $\mathbf{x}_1 - \mathbf{x}_2$ projected from a set of 3-dimensional structures $\mathbf{X}_i$ into these views. Decomposing this matrix yields four possible solutions for rotation and translation, respectively. Only one of the four solutions is physically possible, which can be identified by reconstructing a single 3-dimensional point, and verifying if it is located in front w.r.t. the camera frame. Once the correct combination of rotation and translation is known, the 3-dimensional structure can be reconstructed from the image correspondences by the method of triangulation. The rotation estimation is unique and causes no problems, however, translation as well as the depth of the structure can only be recovered up to a scalar factor which results from the fact that the essential matrix only has a rank of 2. Hence, only the translational direction of the camera can be determined, not the magnitude.

In order to recover the essential matrix and 3-dimensional structure robustly w.r.t. to unavoidable noise, a sufficiently large baseline between two views is necessary, as well as two distinct vantage points. Unfortunately, when working with a quad-rotor UAV both previous requirements are difficult to meet, since this kind of vehicles

normally perform small movements between consecutive images. In addition, their movement is mostly performed in the camera's direction of view. The first problem can be solved by skipping images until the baseline between the evaluated par of images is large enough. Unfortunately, this approach does not meet the UAVs control requirement of having fast position and velocity estimates.

When using a monocular system, the 3-dimensional structure can only be recovered as long as the vehicle is moving. Since it is desirably that the quad-rotor navigates slowly while performing missions, 3-dimensional reconstruction becomes difficult, making very problematic some elementary tasks like obstacle avoidance.

To overcome the scale ambiguity issue when using monocular vision systems, either some previous scene knowledge is necessary, or information from other sensors (range finder, IMU) have to be taken into account. By placing artificial landmarks of well known dimensions in the environment where the UAV performs, the scale ambiguity problem can be solved effectively. However, since UAVs are meant to fly into unknown environments, previous knowledge of the scene is usually not available. On the other hand, incorporating additional sensors yields also additional problems, like having different measurement rates or inaccurate calibration from these sensors w.r.t. the imaging sensor.

Stereo Imaging System A stereo vision system allows to reconstruct 3-dimensional structure as well as motion directly, independently of the motion itself. Placing the pair of cameras in a considerable distance from one to the other, a stereo rig enforces a sufficiently large baseline between both views. In addition, it also allows to reconstruct the 3-dimensional feature position in a single time-step. In addition, stereo vision techniques can deal with the degenerate configuration in which all 3-dimensional features considered lie on a plane.

The stereo vision systems have also some disadvantages that are worth to mention. A stereo vision algorithm is computationally more intensive and demanding than monocular vision algorithms. Having two cameras, all image processing tasks like undistortion, rectification and filtering have to be done twice. In addition, the feature's correspondences have to be performed between consecutive images, but also between the left and right images. Furthermore, some hardware issues must be solved, like image processing bus overload, or effective synchronization of the cameras. If the stereo cameras are not synchronized, and the vehicle moves between the acquisition of the stereo pair, the relative pose of the cameras will change and, as consequence, the extrinsic calibration will not fit any more. This penalizes the 3-dimensional structure reconstruction, leading to inaccurate pose estimates. Considering the fast dynamics of the quad-rotor, synchronized cameras are essential for performing stereo imaging studies.

Implementing a stereo imaging system also implies that more components need to be installed onboard, which increases the weight that the UAV must lift. In order to overcome this situation, the quad-rotor design must be carefully conducted. As a conclusion, light frame and onboard components must be chosen, as well as powerful motors, aiming at increasing the UAV payload capacity.

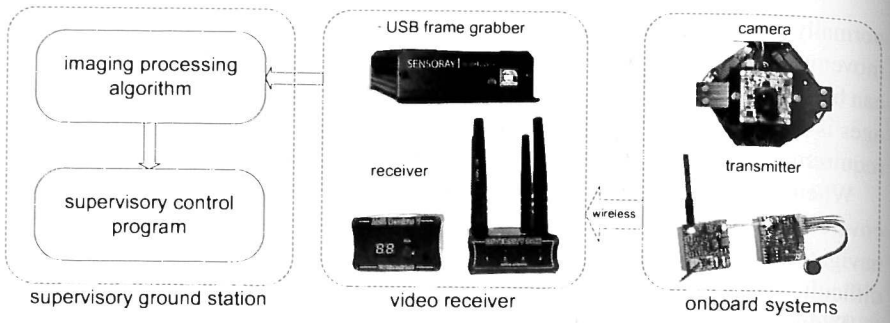


Fig. 5.21 Schema of the monocular vision system

5.4.3 Monocular Imaging System Implementation

The quad-rotor's monocular imaging system was developed with the purpose of enabling the vehicle for performing vision-based tasks. The imaging sensor is a *CTDM-5351* camera from SONY, with a resolution of 640 × 480 pixels, installed in the lower part of the vehicle, placed pointing downwards. The *CTDM-5351* camera has been chosen, since it offers a considerably high quality image within well and poorly illuminated areas. The analog output of the camera is connected directly to a wireless *Micro PLL Transmitter* and a *Micro Booster* of 200 mW. By combining these two components, the transmission power can be improved up to 20 times, ensuring a good quality video during the UAV's missions. With the purpose of reducing possible disturbances in the transmitted images, the video signal is recovered by means of a 4-antenna *Diversity System Receiver*. Incoming signals are received by each one of the four antennas, so that the receiver can evaluate which antenna is providing the most suitable signal. Once detected, the system switches to this antenna automatically. The receiver outputs the video signal via a composite connector to a *Sensoray 2255* USB Frame Grabber, which is specially designed to fast video acquisition. The overall system, composed by transmitter, receiver, and frame grabber, was chosen with the purpose of reducing possible time delays, and to ensure a high quality real-time video.

The frame grabber is connected to the supervisory ground station via a USB port. A C-coded imaging application, based on OpenCV library functions [20], performs image processing for estimating the vehicle's states. OpenCV [64] is a functions library, containing a series of low-overhead, high-performance operations, that can be used to perform fast computing algorithms on images. An overall scheme of the monocular system components is shown Fig. 5.21.

The information extracted by the imaging application is placed on a fixed memory segment that is shared with the supervisory control application. With this method, the supervisory control application receives the required data for performing vision-based tasks.

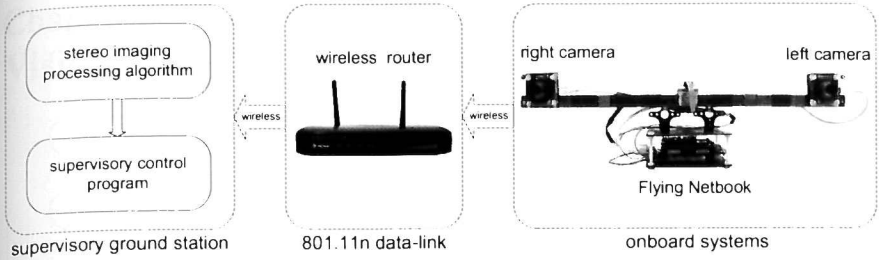


Fig. 5.22 Schema of the stereo imaging system

5.4.4 Stereo Imaging System Implementation

A schema showing the quad-rotor's stereo imaging system can be seen in Fig. 5.22. This arrangement is designed to be installed on the top of the helicopter, positioned facing forward. Two monochromatic *uEye UI-1226LE-M-G* cameras from *IDS GmbH*, having a resolution of 376 × 240 pixels and a field of view of 75° have been chosen as imaging sensors. The cameras are mounted on a carbon rig, separated one from another by a distance of 35 cm. As can be seen in the corresponding image, an additional IMU has been installed in a centered position between both cameras. The purpose of this inertial sensor will be explained later, in Chap. 7. The stereo imaging system has been accurately calibrated as shown previously in Sect. 5.2.2, which allows the evaluation of the epipolar constraint from (5.23).

The cameras operate in an external trigger mode, therefore, every image acquisition has to be requested by sending a hardware trigger signal to each camera. *uEye* cameras are equipped with hardware input/output ports, one of which has been programmed as hardware trigger input. The left camera is enabled as *master*, and one of its input/output ports is configured as an output that is triggered by software. This output is connected to the left as well as to the right camera's trigger input, therefore, both cameras are waiting for the hardware trigger to perform image acquisition.

The cameras are connected via mini-USB ports to an embedded Flying Netbook board, developed by *Ascending Technologies GmbH* [8]. This mini computer counts with an Intel Atom 1.6 GHz processor, which gives enough power for executing a C-coded program based on OpenCV functions, which deals with the next group of tasks. The first step of the program consists of generating the trigger signal required for synchronizing the *uEye* cameras. Once the cameras receive the trigger, a synchronized images pair is captured. The program proceeds to recuperate both images, next, it uses OpenCV functions to compress them in jpeg format. Both images are then stored in a data buffer, which is sent via a 801.11n wireless data-link to the supervisory ground station PC. Communication between the *Flying Netbook* and the supervisory ground station is achieved by using a *TEW-652BRP Wireless Nspeed Broadcast Router* from *TRENDnet*.

Once the data buffer is received, a C-coded application based on OpenCV functions decompresses the images and performs the required imaging processing over the stereo pair. This step provides accurate measurements of the helicopter (x, y, z)

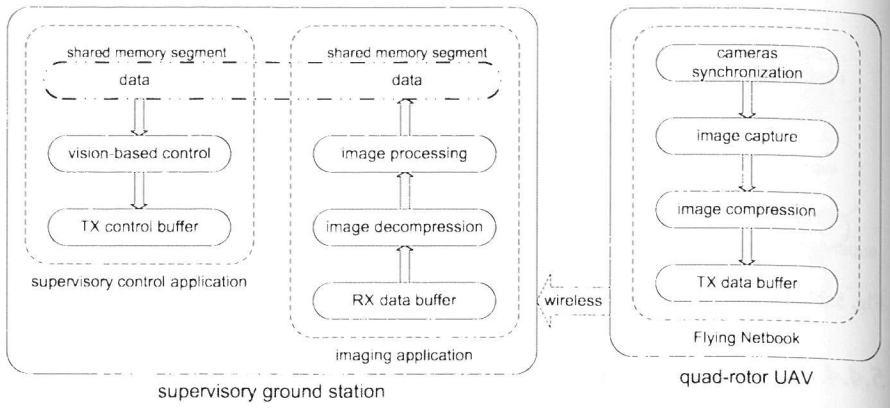


Fig. 5.23 Schema of the stereo imaging process and the vision-based control computation

3-dimensional position relative to its surrounding environment. Such data are placed on a fixed memory segment that is shared with the supervisory control application. With this method, the supervisory control application receives the required data for performing vision-based tasks. A schematic showing the stereo imaging process and the vision-based control computation is shown in Fig. 5.23.

5.5 Concluding Remarks

This chapter provided fundamental background behind the implementation of imaging sensors for estimating relative position and velocity. With the purpose of estimating the internal parameters of a camera, the calibration of a monocular system has been addressed. In addition, a stereo imaging system has been calibrated, which allowed the reconstruction of a 3-dimensional point from its corresponding 2-dimensional projections. Both calibration methods were based on the *Camera Calibration Toolbox for Matlab*. For the computation of relative translational speed using an imaging sensor, the concept of optical flow and a method for its computation have been detailed.

Some important points that must be considered when implementing a monocular or a stereo imaging system onboard a quad-rotor UAV have been analyzed and discussed. This allowed the identification of some benefits and drawbacks inherent to each approach, and motivated the development of both systems for their evaluation in real-time experiments. The components conforming the monocular system, as well as the software architecture allowing the estimation of the data required for performing vision-based tasks were presented. A similar explanation is given for the stereo imaging system.

The following sections of this book show the performance of both systems for estimating the states of a quad-rotor UAV, which allows controlling the flight of the vehicle during flight experiments.

Chapter 6
Vision-Based Control of a Quad-Rotor UAV

This chapter presents two different vision-based strategies for stabilizing a quad-rotor UAV during flight. Theoretical and practical aspects are detailed, as well as the physical set-ups developed, allowing the realization of real-time experimental applications.

The chapter is divided as follows. Section 6.1 presents a vision-based strategy for stabilizing the quad-rotor during flight. This technique is based on a homography estimation technique and an optical flow computation. Next, a comparison of three control strategies is addressed in Sect. 6.2, with the purpose of identifying the most effective approach for stabilizing the vehicle when using visual feedback. In Sect. 6.3 the vision system is implemented for allowing altitude control, with the objective of stabilizing the 3-dimensional position and regulating the velocity of the vehicle using optical flow. Finally, some concluding remarks are presented in Sect. 6.4.

6.1 Position Stabilization Using Vision

6.1.1 Introduction

With the purpose of enhancing the autonomy level of the "Cross-Flyer" UAV presented in Sect. 3.4.1, a vision-based position control approach is proposed. The helicopter's (x, y, z) positions and $(\dot{x}, \dot{y}, \dot{z})$ velocities are estimated with respect to a landing pad on the ground, using the monocular vision system presented in Sect. 5.4.3. The developed technique allows measuring the states, which are difficult to obtain from conventional navigation systems, for example GPS, when performing in urban environments or indoors.

In the proposed approach, the 3-dimensional position estimation is based on the computation of homographies, while translational velocity is obtained based on an optical flow computation. The states estimated are applied in a full state feedback controller, with the purpose of generating control input for stabilizing the quad-rotor

L.R. García Carrillo et al., *Quad Rotorcraft Control*, Advances in Industrial Control,
DOI 10.1007/978-1-4471-4399-4_6, © Springer-Verlag London 2013

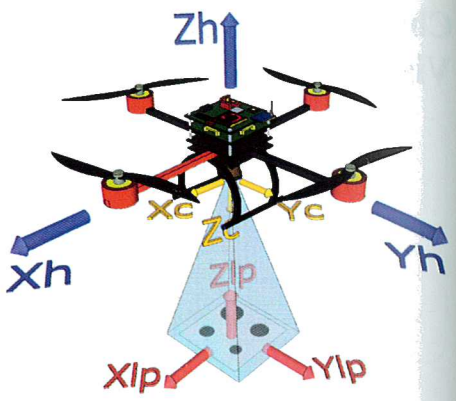


Fig. 6.1 Vision-based position stabilization schema

(x, y, z) position during flight. The effectiveness of the proposed method has been verified under real-time experiments. Some graphics representing the behavior of the helicopter are shown to illustrate the performance of the UAV during flight.

6.1.2 Visual System Set-up

Controlling the 3-dimensional position of a UAV depends on the knowledge of the (x, y, z) vehicle coordinates and $(\dot{x}, \dot{y}, \dot{z})$ translational velocities with respect to a well-known reference frame. Such values are required data for the controller in order to generate the control input to stabilize the aircraft over a desired location. To fulfill this situation, the vision system presented in Sect. 5.4.3 has been implemented, in order to provide the required position and velocities information. The complete system proposed consists of the calibrated camera onboard the UAV, a landing pad or artificial marker placed on ground, an imaging processing algorithm running on a supervisory ground station PC, and the wireless link between the helicopter and the supervisory ground station. Figure 6.1 shows the proposed system which can be described as:

- **Quad-rotor UAV:** with a body fixed frame (X_h, Y_h, Z_h), assumed to be at its center of gravity. Z_h represents the yaw axis, and pointing upwards. X_h and Y_h are the roll and pitch axis, respectively.
- **Strapdown camera:** pointing downwards, with a reference frame (X_c, Y_c, Z_c). When moving, the camera surveys the scene passing below the quad-rotor. Since $X_c - Y_c$ and $X_h - Y_h$ are considered as parallel planes, then, the visual information collected by the camera can be used to stabilize the vehicle.
- **Landing pad:** artificial landmark of known dimensions, formed by four circles of known coordinates, painted on high contrast background and placed underneath the rotorcraft. The coordinates frame (X_{lp}, Y_{lp}, Z_{lp}) represents the inertial reference frame.

The planes formed by $(X_h - Y_h)$ and $(X_{lp} - Y_{lp})$ are considered to be parallel because it is assumed that the rotorcraft is in hover flight over the landing pad. Running in the supervisory ground station, an algorithm for imaging processing provides an estimate of the helicopter altitude, position in the (X, Y) plane, and translational velocities. As mentioned earlier, the 3-dimensional position information is deduced by an homography estimation technique applied to the image of the landing pad, while the translational velocities are estimated by means of optical flow computation. All information sensed by the camera is related to the landing pad image. It will be shown that this information is rich enough to stabilize the 3-dimensional position of the vehicle. Let us explain first the procedure performed to compute the vehicle's positions and velocities.

6.1.3 Vision-Based Position Estimation

This section describes the imaging algorithms developed with the purpose of estimating the vehicle's states required for stabilizing its position during flight. First, a homography estimation technique for computing the 3-dimensional position is described, followed by a technique for deriving translational velocities. Finally, a method for estimating the homography when the detection of the landing pad fails is presented.

6.1.3.1 Computing the 3-dimensional Position

In order to estimate the UAV position relative to the landing pad, the extrinsic parameters of the camera are computed at every image frame. This is achieved by implementing a homography estimation technique, which provides the (x, y, z) position and (ψ, θ, ϕ) orientation of the camera with respect to the artificial landmark in the image scene. The action of the homography can be expressed as [20]

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = sK \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{T} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{6.1}$$

where $[x\ y\ 1]^{\mathrm{T}}$ represents the landing pad position in the camera image, s is a known scale factor, $K \in \mathbb{R}^{3\times3}$ represents the intrinsics parameters camera matrix (previously found in (5.19)), $R = [\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{r}_3] \in \mathbb{R}^{3\times3}$ are the extrinsic rotation parameters, $\mathbf{T} \in \mathbb{R}^{3\times1}$ is the extrinsic translation parameters vector and $[X\ Y\ Z\ 1]^{\mathrm{T}}$ is the real landing pad position.

Without loss of generality, the landing pad plane is defined so that $Z = 0$. This is done because, if one also breaks up the rotation matrix into three 3×1 columns (i.e., $R = [r_1\ r_2\ r_3]$), then one of those columns is not needed. Therefore, the homography matrix H can be expressed as $H = sK[r_1\ r_2\ \mathbf{T}]$. The homography matrix

H is divided in two parts: the physical transformation (which locates the observed object plane) and the projection (the camera intrinsic matrix).

Rotation R is described by three angles and translation $\mathbf{T}$ is defined by three offsets; hence there are six unknowns for each view. The known planar object (the artificial landmark) provides eight equations, that is, the mapping of a rectangle into a quadrilateral can be described by four (x, y) image points. For every instant, when the aerial vehicle is in hovering, it is possible to compute the homography matrix H using the a priori knowledge of the position of the four centroids of the circles [44]. Using this estimated transformation matrix and the intrinsic camera matrix previously identified by an off-line calibration based on the method in [17], it is possible to calculate the camera extrinsic parameters, and therefore one has the vehicle's (x, y, z) position with respect to the landing pad on the ground.

Each one of the circles are detected in the image using an OpenCV function, next, they are classified according to the magnitude of its radius, allowing a correct identification of the landing pad orientation. The first circle corresponds to the upper left circle (or circle with smallest radius), continuing with the upper right circle as the second in the list. The lower left circle comes next according to its radius, and finally, the lower right circle is identified as the circle whose radius magnitude is bigger. Figure 6.2 shows an image of the landing pad, as viewed from the camera, where the four circles perimeters are highlighted with different colors according to the magnitude of its radius.

An erroneous detection of the landing pad circles must be discarded, since it will provide an erroneous position estimation. With this purpose, the parallelism of the lines mapped from the landing pad must be verified. Figure 6.1 shows that the four circles of the landing pad are positioned forming the corners of a rectangle. The line between the two upper corners and the line joining the two lower corners must satisfy a parallelism constrain. The same restriction is checked for the line joining the two left corners and the line between the two right corners. Parallelism verification is based on the slope of a line equation:

$$m_* = \frac{y_f - y_i}{x_f - x_i} \tag{6.2}$$

where i and f stand for initial and final coordinates, respectively. Thus, the slope m_{up} of the upper line must be almost equal to the slope m_{lo} of the lower line, while the slope m_{le} of the left line must be almost equal to the slope m_{ri} of the right line:

$$|m_{\text{up}} - m_{\text{lo}}| < \varepsilon; \qquad |m_{\text{le}} - m_{\text{ri}}| < \varepsilon \tag{6.3}$$

where ε stands for a constrain helping to determine the lines parallelism. Every detection of the four circles, validating the two previous verifications, ensures that a good planar homography could be estimated, resulting in a good computation of the camera extrinsic parameters. The four lines between circles, defining the sides of the rectangle, are highlighted in Fig. 6.2.

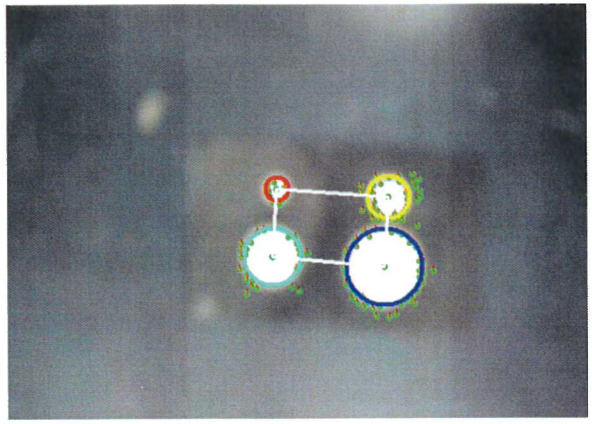


Fig. 6.2 Image processing for homography estimation: Detection of the four circles and parallel lines verification

6.1.3.2 Translational Velocities

An optical flow computation procedure is applied to compute the $(\dot{x}, \dot{y}, \dot{z})$ translational velocities of the aerial vehicle with respect to the landing pad. In order to compute optical flow, the Lucas–Kanade pyramidal algorithm [16] has been implemented in combination with a feature-detecting algorithm. This approach provides an accurate estimation of the motion field since it does not take into account the non landing pad areas, where the motion field cannot be accurately determined.

Let us consider the camera moving with respect to a rigid scene. The velocities and rotation rates of the camera in the inertial frame are expressed by (V_x, V_y, V_z) and (w_x, w_y, w_z), respectively. To accurately estimate the pseudo-speeds of the engine, let us define a tracking zone surrounding the landing pad, in a way that the centroid of the zone and the center of the landing pad coincide. The most representative features over the zone are detected using an OpenCV function devoted to such task. These features, usually the circle perimeters, are selected as *features to track for*. A tracking process based on OpenCV-based optical flow computation is performed over the entire image, allowing measuring the displacements of the tracked features.

Thus, the mean of the optical flow computed on all those points can be expressed as a function of the camera movement as follows:

$$\bar{\text{OF}}_x = \bar{V}_{\text{OF}_x} + K_x \bar{V}_{\text{OF}_z} + \bar{R}_{\text{OF}_x} \tag{6.4}$$

$$\bar{\text{OF}}_y = \bar{V}_{\text{OF}_y} + K_y \bar{V}_{\text{OF}_z} + \bar{R}_{\text{OF}_y} \tag{6.5}$$

Using the results presented in [68], the rotational optical flow is compensated and the pseudo-speeds $(\bar{V}_{\text{OF}_x}, \bar{V}_{\text{OF}_y}, \bar{V}_{\text{OF}_z})$ are deduced. Since the camera system and the helicopter share the same movements, it can be said that the deduced pseudo-velocities depend of the rotorcraft movement. Indeed, the camera is mounted onboard the quad-rotor and fixed in a way it has no freedom degree. Thus, we may write

$$\bar{V}_{\mathrm{OF}_x} = -\frac{f\dot{x}}{z} \tag{6.6}$$

$$\bar{V}_{\mathrm{OF}_y} = -\frac{f\dot{y}}{z} \tag{6.7}$$

$$\bar{V}_{\mathrm{OF}_z} = \frac{\dot{z}}{z} \tag{6.8}$$

where $(\dot{x}, \dot{y}, \dot{z})$ is the speed vector of the rotorcraft center of gravity and z is the altitude. Thus, from these three equations the proposed optical flow vision system allows speed estimation of the rotorcraft up to a scale factor, when flying at constant altitude. Those estimations can be used to control the translational velocities of the rotorcraft.

6.1.3.3 Prediction of the Landing Pad Position

Changes of illumination between one frame to another as well as occlusions can lead to a poor performance of imaging processing algorithms. Due to this, if the landing pad is not successfully detected in the current image, the vision algorithm will fail. To overcome such situation, optical flow measurements are applied to estimate the position of the four circles centroids in the current image as

$$\rho_x^k = \rho_x^{k-1} + \Delta_T \bar{V}_{\mathrm{OF}_x} \tag{6.9}$$

$$\rho_y^k = \rho_y^{k-1} + \Delta_T \bar{V}_{\mathrm{OF}_y} \tag{6.10}$$

where (ρ_x^k, ρ_y^k) represents the circle's centroid position at time instant k, and Δ_T is the working frequency of the algorithm. From this is evident the benefit obtained from applying optical flow for computing the translational velocities. The centroid positions estimated from (6.9)–(6.10) allow to compute the homography at each frame in case that the detection of the landing pad fails.

6.1.4 Control Strategy

Since there exists a wireless communication between the helicopter and the supervisory ground station, the hierarchical controller presented in Sect. 3.5 can be implemented to perform deported vision-based stabilization tasks. The control scheme for the overall cascaded system is then formed by:

- **Position control:** a high-level control running in the supervisory ground station.
- **Attitude control:** a low-level control, performed by the autopilot embedded in the vehicle.

The imaging algorithm described in Sect. 6.1.3 provides the vehicle's states required in the control strategy. Such information is shared between the supervisory control program and the imaging application by means of a shared memory, as explained previously in Sect. 3.3.

6.1.4.1 Altitude and Yaw Control

The control of the vertical position (2.32) can be obtained by using the following control input:

$$u = (-k_{vz}\dot{z} - k_{pz}e_z + mg)\frac{1}{\cos\theta\cos\phi} \tag{6.11}$$

with $e_z = z - z_{\mathrm{d}}$ as the z error position and z_{d} as the desired altitude. k_{pz} and k_{vz} are positive constants. Thus, for the altitude dynamics, r_1 is a PD controller. In the case of the yaw angular position (2.33), one can apply

$$\tilde{\tau}_\psi = -k_{v\psi}\dot{\psi} - k_{p\psi}e_\psi \tag{6.12}$$

where $e_\psi = \psi - \psi_{\mathrm{d}}$ denotes the yaw error, ψ_{d} represents the desired yaw angle, $k_{p\psi}$ and $k_{v\psi}$ denote the positive constants of a PD controller. Indeed, introducing (6.11) and (6.12) into the set of equations (2.30)–(2.35) and provided that $\cos\theta\cos\phi \neq 0$, one has

$$m\ddot{x} = (-k_{vz}\dot{z} - k_{pz}e_z + mg)\left(\frac{\sin\psi\tan\phi}{\cos\theta} + \cos\psi\tan\theta\right) \tag{6.13}$$

$$m\ddot{y} = (-k_{vz}\dot{z} - k_{pz}e_z + mg)\left(\sin\psi\tan\theta - \frac{\cos\psi\tan\phi}{\cos\theta}\right) \tag{6.14}$$

$$m\ddot{z} = -k_{vz}\dot{z} - k_{pz}e_z \tag{6.15}$$

$$\ddot{\psi} = -k_{v\psi}\dot{\psi} - k_{p\psi}e_\psi \tag{6.16}$$

The control parameters $k_{p\psi}$, $k_{v\psi}$, k_{pz}, and k_{vz} should be carefully chosen to ensure a stable well-damped response in the vertical and yaw axes [27]. From (6.15) and (6.16) it follows that $\psi \to \psi_{\mathrm{d}}$ and $z \to z_{\mathrm{d}}$. From (6.11)–(6.12) and (6.15)–(6.16) $r_1 \to 0$ and $\psi \to \psi_{\mathrm{d}}$. For a time T large enough, e_z and e_ψ are arbitrarily small, therefore, (6.13) and (6.14) reduce to

$$\ddot{x} = g\tan\theta \tag{6.17}$$

$$\ddot{y} = -g\frac{\tan\phi}{\cos\theta} \tag{6.18}$$

6.1.4.2 Control of Forward Position and Pitch Angle

Consider the subsystem given by (2.34) and (6.17). Let us impose a very small upper bound on $|\theta|$ in such a way that the difference $\tan(\theta) - \theta$ is arbitrarily small ($\theta \approx \tan(\theta)$). Therefore, the subsystem (2.34) and (6.17) becomes the following linearized system:

$$\ddot{x} = g\theta \tag{6.19}$$

$$\ddot{\theta} = \tilde{\tau}_\theta \tag{6.20}$$

The longitudinal subsystem focuses on the x–θ stabilization. The goal is to control the pitch angle θ and the longitudinal position x to desired references. Consider

the linearized system (6.19)–(6.20), the longitudinal position of the vehicle can be described as a chain of integrators,

$$\dot{x}_1 = x_2 \tag{6.21}$$
$$\dot{x}_2 = x_3 \tag{6.22}$$
$$\dot{x}_3 = x_4 \tag{6.23}$$
$$\dot{x}_4 = \tilde{\tau}_\theta \tag{6.24}$$

where x_1 is the error between the desired and the actual position, $x_3 = \theta$ and $x_4 = \dot{\theta}$. The hierarchical control is constructed by separating, in the previous subsystem, the rotational dynamics from the translational dynamics (low-level control and position control, respectively). Considering x_3 as the control input for the translational dynamics, and applying a backstepping change of variables, one has

$$\dot{x}_1 = x_2 \tag{6.25}$$
$$\dot{x}_2 = x_3^{\text{ref}} + \tilde{x}_3 \tag{6.26}$$
$$\dot{\tilde{x}}_3 = \tilde{x}_4 \tag{6.27}$$
$$\dot{\tilde{x}}_4 = \ddot{x}_2^{\text{ref}} + \tilde{\tau}_\theta \tag{6.28}$$

Here $x_3^{\text{ref}} = \theta_{\text{ref}}$, $\tilde{x}_3 = \theta - \theta_{\text{ref}}$, and $\tilde{x}_4 = \dot{\theta} - \dot{\theta}_{\text{ref}}$. θ_{ref} is the reference angle chosen to control the longitudinal displacement. Notice that setting the reference pitch angle to zero equals to stop the latter displacement. Since the low-level (rotational) control runs at highest rate than the navigation control, the reference angle dynamics can be ignored, then ($\dot{\theta}_{\text{ref}} = \ddot{\theta}_{\text{ref}} = 0$). Indeed, the dynamics of θ will converge faster than the position control. Then, one has

$$\dot{x}_1 = x_2 \tag{6.29}$$
$$\dot{x}_2 = x_3^{\text{ref}} + \tilde{x}_3 \tag{6.30}$$
$$\dot{\tilde{x}}_3 = x_4 \tag{6.31}$$
$$\dot{x}_4 = \tilde{\tau}_\theta \tag{6.32}$$

The longitudinal position is stabilized by defining the reference pitch angle as a function of the longitudinal position and velocity

$$\theta_{\text{ref}} = V_x(x, \dot{x}) = -k_d^x \dot{x} - k_p^x (x - x_d) \tag{6.33}$$

which can be defined as PD controller. The pitch reference angle is used in the rotational control as

$$\tilde{\tau}_\theta = -k_v^\theta x_4 - k_p^\theta \tilde{x}_3 \tag{6.34}$$
$$\tilde{\tau}_\theta = -k_v^\theta \dot{\theta} - k_p^\theta (\theta - \theta_{\text{ref}}) \tag{6.35}$$

The controller gains k_v and k_p should be chosen appropriately so that the polynomial $s^2 + k_v s + k_p$ is Hurwitz.

6.1.4.3 Control of Lateral Position and Roll Angle

Consider the subsystem given by (2.35) and (6.18). Let us impose a very small upper bound on $|\phi|$ in such a way that the difference $\tan(\phi) - \phi$ is arbitrarily small ($\phi \approx \tan(\phi)$). Therefore, the subsystem (2.35) and (6.18) becomes

$$\ddot{y} = -g\phi \tag{6.36}$$
$$\ddot{\phi} = \tilde{\tau}_\phi \tag{6.37}$$

The translational subsystem focuses on the y–ϕ stabilization. Consider the linearized system equations (6.36)–(6.37), the translational position of the vehicle can be described as a chain of integrators

$$\dot{y}_1 = y_2 \tag{6.38}$$
$$\dot{y}_1 = y_2 \tag{6.39}$$
$$\dot{y}_2 = y_3 \tag{6.40}$$
$$\dot{y}_3 = \tilde{\tau}_\phi \tag{6.41}$$

Considering ϕ as the control input for the translational dynamics, then the chain of four integrators can be separated into two subsystems composed of two integrators for both of them. The implementation of a backstepping change of variables yields

$$\dot{y}_1 = y_2 \tag{6.42}$$
$$\dot{y}_2 = y_3^{\text{ref}} + \tilde{y}_3 \tag{6.43}$$
$$\dot{\tilde{y}}_3 = y_4 \tag{6.44}$$
$$\dot{y}_4 = \tilde{\tau}_\phi \tag{6.45}$$

where $y_3^{\text{ref}} = \phi_{\text{ref}}$. Using high gains in the low-level attitude control, the dynamics of the reference angle can be ignored. The lateral position is then stabilized by defining the reference roll angle as a function of the lateral position and velocity,

$$\phi_{\text{ref}} = V_y(y, \dot{y}) = -k_d^y \dot{y} - k_p^y (y - y_{\text{d}}) \tag{6.46}$$

which can be defined as PD controller. The roll reference angle is used in the rotational control as

$$\tilde{\tau}_\phi = -k_v^\phi y_4 - k_p^\phi \tilde{y}_3 \tag{6.47}$$
$$\tilde{\tau}_\phi = -k_v^\phi \dot{\phi} - k_p^\phi (\phi - \phi_{\text{ref}}) \tag{6.48}$$

The controller gains k_v and k_p should be chosen appropriately so that the polynomial $s^2 + k_v s + k_p$ is Hurwitz.

Fig. 6.3 The four-rotor aircraft experimental platform with the camera pointing downwards

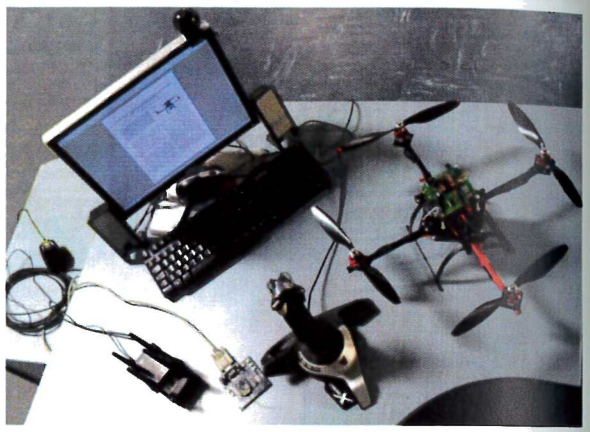


Fig. 6.4 The experimental system configuration: "Cross-Flyer" UAV and supervisory ground station

6.1.5 Experimental System Configuration

The imaging algorithms presented in Sect. 6.1.3 and the hierarchical controller presented in Sect. 6.1.4 have been tested over a system composed by the "Cross-Flyer" UAV platform, a supervisory ground station, and a wireless video link (onboard transmitter, on ground receiver).

The "Cross-Flyer" UAV with the downward looking camera is shown in Fig. 6.3. The supervisory ground station shown in Fig. 3.2 was used for these studies. It consists of a desktop PC, the joystick and the XBee ZB ZigBee PRO radio modem, and the 4-antenna *Diversity System Receiver*. The ground station runs a supervisory control application allowing a user to send information to the helicopter and to chose between a manual control or an autonomous vision-based position hold. The supervisory ground station receives and saves data sent by the vehicle in order to debug and analyze the flight experiments. The control feedback between the supervisory ground station and the helicopter is performed at 15 Hz. The complete experimental system can be seen in Fig. 6.4.

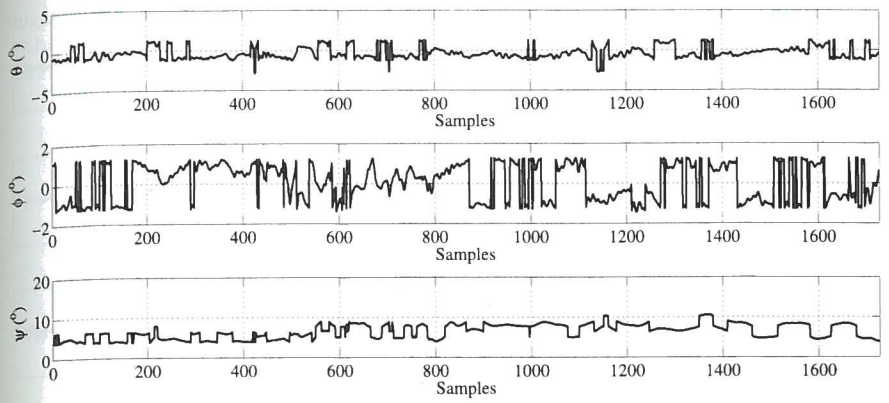


Fig. 6.5 Euler angles, experimental results

6.1.6 Experimental Applications

With the purpose of validating the vision-based control strategy proposed, a set of experiments were conducted. The goal consists of stabilizing the 3-dimensional position of the quad-rotor during flight, with respect to the landing pad placed on the ground. During the tests, the vision system provides the position and velocity feedback, while the embedded inertial electronic system provides the attitude data. The experiment's procedure is as follows.

First, the UAV is positioned exactly on top of the landing pad. Next, the operator uses the joystick in the supervisory ground station to define the current vehicle's position as the desired (x_d, y_d) position reference. The desired altitude z_d is always fixed at 150 cm, and the desired yaw angle ψ is fixed at 5° degrees. Using such values, the control strategy deals with stabilizing the vehicle during experiments. The control parameters used are: $k_{pz} = 0.68$, $k_{vz} = 1.6$, $k_{p\psi} = 38$, $k_{v\psi} = 1350$, $k_{px} = 1$, $k_{vx} = 2$, $k_p^{\theta} = 38$, $k_v^{\theta} = 1350$, $k_{py} = 1$, $k_{vy} = 2$, $k_p^{\phi} = 38$, $k_v^{\phi} = 1400$. As a reminder, the attitude stabilization control system is always running at higher frequency in order to guarantee that the Euler angles are close to zero (hover flight).

The set of Figs. 6.5, 6.6, and 6.7 show the results obtained from the experiment. In those figures, each 10 samples are equivalent to 1 second. Notice that the pitch and roll angles remain in the interval $(-1.5°, 1.5°)$ degrees. Therefore, it can be concluded that the position control adds only small changes in the attitude of the rotorcraft for bringing the position to the desired one. This is an important property because the position controller runs at a lower rate compared to the attitude controller, then, a smooth position control is necessary to ensure the stability of the vehicle. A picture of the quad-rotor during a real-time experiment is depicted in Fig. 6.8. It can be seen that the UAV maintains its position relative to the landmark. A video of the quad-rotor while performing experiments can be seen at http://www.youtube.com/watch?v=SQlSXruTnj0.

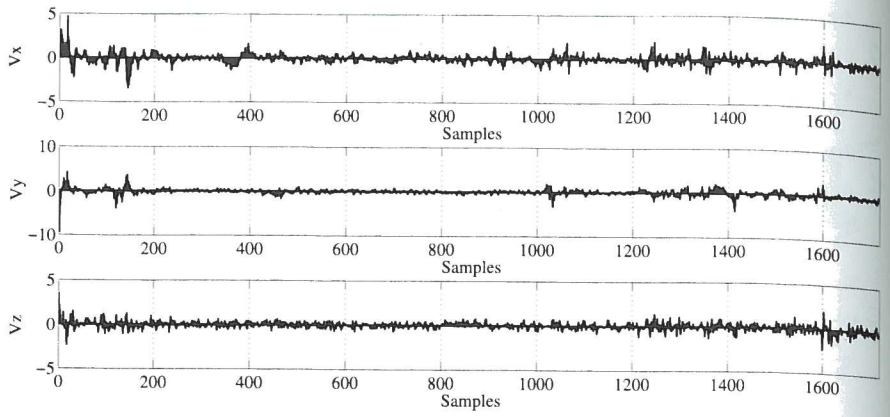


Fig. 6.6 Velocities, experimental results

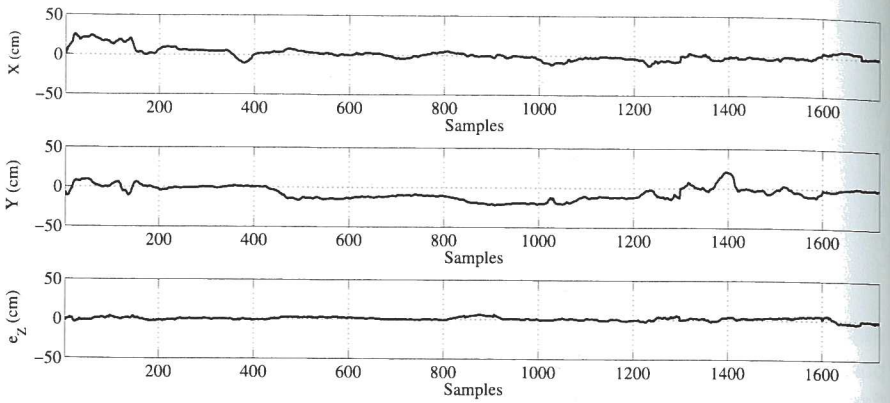


Fig. 6.7 Positions, experimental results

6.1.7 Final Comments

In this section, a vision-based strategy for stabilizing the 3-dimensional position of a quad-rotor with respect to a landing pad on the ground was proposed and tested in real-time experiments. The proposed vision algorithm consists of an homography estimation technique for extracting the 3-dimensional position, as well as of an optical flow computation for deriving the vehicle's translational velocities. A technique for predicting the landing pad position when its detection fails was presented also. The estimated information proved to be rich enough to allow performing autonomous missions. The experimental application was successfully performed indoors showing that the quad-rotor was stabilized at a selected (x, y, z) position above the landing pad. The attitude of the vehicle was not significantly perturbed by the vision-based control input used to correct the UAV position. The vehicle's velocities remained also very close to zero.

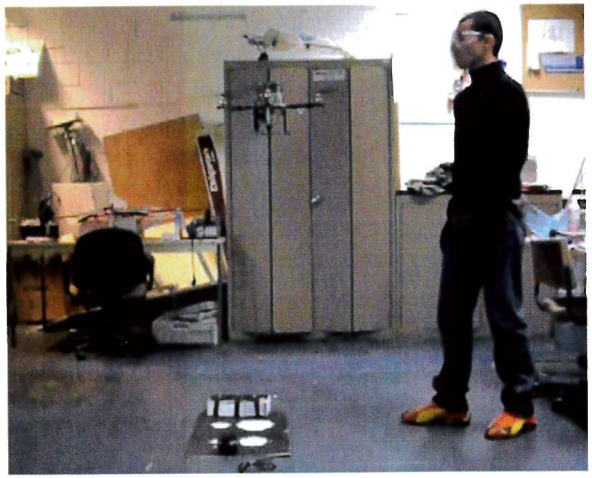


Fig. 6.8 The quad-rotor stabilized over the landing pad in a desired position

6.2 A Comparison of Nonlinear Controllers Using Visual Feedback

The previous section presented a vision-based stabilization of the quad-rotor using a hierarchical control strategy. Although the experimental results obtained demonstrate the effectiveness of such approach, it was considered that evaluating different control strategies in real-time experiments will help to determine the better control strategy for the research objectives. This reasoning motivated the present studies.

6.2.1 Introduction

Testing the performance of different controls over quad-rotors systems is a subject already studied. PID and LQR controllers are compared in [14], while in [15], the performance of a backstepping and a sliding modes controllers are tested. For the last two examples, experiments were performed over a quad-rotor platform, where three degrees of freedom are locked. The authors conclude that backstepping control technique is the most appropriate approach for their future work. In [6], two control methods are studied over a quad-rotor platform equipped with visual feedback. These methods are based on feedback linearization and a backstepping-like control.

The control objective of the studies presented in this Section consists of identifying the most effective controller for stabilizing the position of the "Cross-Flyer" UAV with respect to an artificial visual landmark placed on the ground. Three controllers considered between the most commonly reported in the literature have been chosen: nested saturations [27, 77], backstepping [70], and sliding modes [52]. The performance of such methodologies applied to the quad-rotor system is evaluated from real-time experimental results.

6.2.2 System Set-up

The vision-based position estimation approach used for these experiments is based on the same visual system set-up presented previously in Sect. 6.1.2. Similarly, visual feedback is provided from the imaging algorithms presented in Sect. 6.1.3, and the whole system composed by the "Cross-Flyer" UAV and supervisory ground station is the one presented in Sect. 6.1.5.

6.2.3 Control Strategies

The three control strategies applied to the quad-rotor stabilization are: nested saturations control method, backstepping approach, and sliding modes controller. The three of them are designed to stabilize the x, y, θ and ϕ states. The helicopter's altitude z and yaw angle ψ are stabilized by PD controllers, as shown in Sect. 6.1.4.1. The control algorithms were implemented onboard to stabilize the UAV's 3-dimensional position and attitude.

6.2.3.1 Nested Saturations Control

Consider a system given by four integrators in cascade:

$$\dot{x}_1 = \alpha x_2, \qquad \dot{x}_2 = \beta x_3, \qquad \dot{x}_3 = \gamma x_4, \qquad \dot{x}_4 = u \tag{6.49}$$

where α, β, $\gamma \neq 0$ are constants. A nested saturations control input can be defined as [27, 77]:

$$u = -\sigma_{b_4}\left(k_4 z_4 + \sigma_{b_3}\left(k_3 z_3 + \sigma_{b_2}\left(k_2 z_2 + \sigma_{b_1}(k_1 z_1)\right)\right)\right) \tag{6.50}$$

where z_i, for $i = 1, \ldots, 4$, denotes a change of variables. $k_i > 0$ are constants, and σ_{b_i} represent saturation functions:

$$\sigma_{b_i}(s) = \begin{cases} -b_i; & s < -b_i \\ s; & |s| \leq b_i \\ b_i; & s > b_i \end{cases} \tag{6.51}$$

where $b_i > 0$ are constants denoting the bounds of the saturation functions. The z_i terms are given by

$$z_1 = x_4 + \frac{k_4 + k_3 + k_2}{\gamma} x_3 + \frac{k_3 k_4 + k_2 k_3 + k_2 k_4}{\beta\gamma} x_2 + \frac{k_2 k_3 k_4}{\alpha\beta\gamma} x_1 \tag{6.52}$$

$$z_2 = x_4 + \frac{k_4 + k_3}{\gamma} x_3 + \frac{k_3 k_4}{\beta\gamma} x_2 \tag{6.53}$$

$$z_3 = x_4 + \frac{k_4}{\gamma} x_3 \tag{6.54}$$

$$z_4 = x_4 \tag{6.55}$$

Note that from (6.11)–(6.12) and (6.15)–(6.16) $r_1 \to 0$ and $\psi \to \psi_d$. For a time T large enough, e_z and e_ψ are arbitrarily small, therefore, (6.13) and (6.14) reduce to

$$\ddot{x} = g \tan\theta \tag{6.56}$$

$$\ddot{y} = -g \frac{\tan\phi}{\cos\theta} \tag{6.57}$$

Control of Forward Position and Pitch Angle Consider the subsystem given by (2.34) and (6.56). Implementing a nonlinear control based on nested saturations allows in the limit a guarantee of arbitrary bounds for x, $\dot{x}$, θ and $\dot{\theta}$. Let us impose a very small upper bound on $|\theta|$ in such a way that the difference $\tan(\theta) - \theta$ is arbitrarily small ($\theta \approx \tan(\theta)$). Therefore, the subsystem (2.34) and (6.56) becomes the following linearized system:

$$\ddot{x} = g\theta \tag{6.58}$$

$$\ddot{\theta} = \tilde{\tau}_\theta \tag{6.59}$$

which represents four integrators in cascade. Then, by using (6.49)–(6.50) the controller is

$$\tilde{\tau}_\theta = -\sigma_{b_4}\left(\dot{\theta} + \sigma_{b_3}\left(\dot{\theta} + \theta + \sigma_{b_2}\left(\dot{\theta} + 2\theta + \frac{\dot{x}}{g} + \sigma_{b_1}\left(\dot{\theta} + 3\theta + 3\frac{\dot{x}}{g} + \frac{x}{g}\right)\right)\right)\right) \tag{6.60}$$

This control law comes from the technique based on nested saturations control developed in [77]. It is proved in [27] that θ, $\dot{\theta}$, x and $\dot{x}$ converge to zero. To regulate x around a desired position, the most inner term (associated to σ_{b_1}) must be written as $\dot{\theta} + 3\theta + 3\frac{\dot{x}}{g} + \frac{e_x}{g}$, where e_x is the position error, expressed as $e_x = x - x_d$. Here, x_d represents the desired position reference for x.

Control of Lateral Position and Roll Angle Consider the subsystem given by (2.35) and (6.57). Let us impose a very small upper bound on $|\phi|$ in such a way that the difference $\tan(\phi) - \phi$ is arbitrarily small ($\phi \approx \tan(\phi)$). Therefore, the subsystem (2.35) and (6.57) becomes

$$\ddot{y} = -g\phi \tag{6.61}$$

$$\ddot{\phi} = \tilde{\tau}_\phi \tag{6.62}$$

Using a similar procedure to the one proposed for the pitch control, we obtain

$$\tilde{\tau}_\phi = -\sigma_{b_4}\left(\dot{\phi} + \sigma_{b_3}\left(\dot{\phi} + \phi + \sigma_{b_2}\left(\dot{\phi} + 2\phi - \frac{\dot{y}}{g} + \sigma_{b_1}\left(\dot{\phi} + 3\phi - 3\frac{\dot{y}}{g} - \frac{y}{g}\right)\right)\right)\right) \tag{6.63}$$

In order to regulate y around a desired position, the most inner term (associated to σ_{b_1}) must be written as $\dot{\phi} + 3\phi - 3\frac{\dot{y}}{g} - \frac{e_y}{g}$, where e_y is the position error, expressed as $e_y = y - y_d$. Here, y_d represents the desired position reference for y.

6.2.3.2 Backstepping Control

The backstepping technique provides a systematic method to obtain a control law from a chain of integrators. This methodology was introduced in [70].

Control of Forward Position and Pitch Angle Rewrite subsystem given by (6.58)–(6.59) as

$$\dot{x}_1 = x_2 \tag{6.64}$$
$$\dot{x}_2 = g\theta_1 \tag{6.65}$$
$$\dot{\theta}_1 = \theta_2 \tag{6.66}$$
$$\dot{\theta}_2 = \tilde{\tau}_{B\theta} \tag{6.67}$$

where $\tilde{\tau}_{B\theta}$ will define the final backstepping control input. In order to obtain this control input, consider each equation as a new subsystem, where the next state is taken as the input and it is defined as a virtual control to stabilize such a subsystem. For the present case let us start with

$$\dot{x}_1 = x_2 \tag{6.68}$$
$$\zeta_1 = x_1 \tag{6.69}$$

where x_2 represents the input and ζ_1 the output. Let us propose a positive definite function $V_1 = \frac{1}{2}x_1^2$, whose time derivative is given by

$$\dot{V}_1 = x_1\dot{x}_1 = x_1x_2 \tag{6.70}$$

and consider a virtual input $\alpha_1 = (x_2)^v = -k_1x_1$, where k_1 is a positive constant. Then $\dot{V}_1 = -k_1x_1^2$. Now, let ζ_2 be the new output:

$$\zeta_2 = x_2 - \alpha_1 \tag{6.71}$$

The new subsystem to be stabilized is written as

$$\dot{x}_1 = \zeta_2 + \alpha_1 \tag{6.72}$$
$$\dot{\zeta}_2 = g\theta_1 - \dot{\alpha}_1 \tag{6.73}$$

and let us propose a positive definite function $V_2 = V_1 + \frac{\zeta_2^2}{2}$, then

$$\dot{V}_2 = \dot{V}_1 + \zeta_2\dot{\zeta}_2 = \dot{V}_1 + \zeta_2(g\theta_1 - \dot{\alpha}_1) \tag{6.74}$$

Define a virtual input $\alpha_2 = (g\theta_1) = -k_2\zeta_2 + \dot{\alpha}_1$, where k_2 is a positive constant. Then, $\dot{V}_2 = \dot{V}_1 - k_2\zeta_2^2$. Now, let ζ_3 be a new output:

$$\zeta_3 = g\theta_1 - \alpha_2 \tag{6.75}$$

The new subsystem to be stabilized is written as

$$\dot{x}_1 = \zeta_2 + \alpha_1 \tag{6.76}$$
$$\dot{\zeta}_2 = g\theta_1 - \dot{\alpha}_1 \tag{6.77}$$
$$\dot{\zeta}_3 = g\theta_2 - \dot{\alpha}_2 \tag{6.78}$$

and let us propose the positive definite function $V_3 = V_2 + \frac{\zeta_3^2}{2}$, then

$$\dot{V}_3 = \dot{V}_2 + \zeta_3\dot{\zeta}_3 = \dot{V}_2 + \zeta_3(g\theta_2 - \dot{\alpha}_2) \tag{6.79}$$

Define a virtual input $\alpha_3 = (g\theta_2) = -k_3\zeta_3 + \dot{\alpha}_2$, where k_3 is a positive constant. Then, $\dot{V}_3 = \dot{V}_2 - k_3\zeta_3^2$. Let ζ_4 be the new output:

$$\zeta_4 = g\theta_2 - \alpha_3 \tag{6.80}$$

The new subsystem to be stabilized is written as

$$\dot{x}_1 = \zeta_2 + \alpha_1 \tag{6.81}$$
$$\dot{\zeta}_2 = g\theta_1 - \dot{\alpha}_1 \tag{6.82}$$
$$\dot{\zeta}_3 = g\theta_2 - \dot{\alpha}_2 \tag{6.83}$$
$$\dot{\zeta}_4 = g\dot{\theta}_2 - \dot{\alpha}_3 \tag{6.84}$$

and let us propose the Lyapunov candidate function $V_4 = V_3 + \frac{\zeta_4^2}{2}$, then

$$\dot{V}_4 = \dot{V}_3 + \zeta_4\dot{\zeta}_4 = \dot{V}_3 + \zeta_4(g\tilde{\tau}_{B\theta} - \dot{\alpha}_3) \tag{6.85}$$

Let us propose the backstepping control input $\tilde{\tau}_{B\theta}$ as

$$\tilde{\tau}_{B\theta} = \frac{1}{g}(-k_4\zeta_4 + \dot{\alpha}_3) \tag{6.86}$$

where k_4 is a positive constant. Then

$$\dot{V}_4 = \dot{V}_3 - k_4\zeta_4^2 = -k_1x_1^2 - k_2\zeta_2^2 - k_3\zeta_3^2 - k_4\zeta_4^2 \tag{6.87}$$

With the proposed backstepping control input $\tilde{\tau}_{B\theta}$, one has $\dot{V}_4 < 0$, and then the system (6.64)–(6.67) is globally asymptotically stable. In order to express $\tilde{\tau}_{B\theta}$ as a function of x_1, x_2, θ_1 and θ_2, ζ_4 and $\dot{\alpha}_3$ must be rewritten as a function of such variables:

$$\zeta_4 = g\theta_2 + (k_1 + k_2 + k_3)g\theta_1 + (k_1k_2 + k_1k_3 + k_2k_3)x_2 + k_1k_2k_3x_1 \tag{6.88}$$
$$\dot{\alpha}_3 = -(k_1 + k_2 + k_3)g\theta_2 - (k_1k_2 + k_1k_3 + k_2k_3)g\theta_1 - k_1k_2k_3x_2 \tag{6.89}$$

The final control input is rewritten as

$$\tilde{\tau}_{B\theta} = -\frac{\overline{k}_1}{g}x_1 - \frac{\overline{k}_2}{g}x_2 - \overline{k}_3\theta_1 - \overline{k}_4\theta_2 \tag{6.90}$$

where

$$\overline{k}_1 = k_1k_2k_3k_4 \tag{6.91}$$
$$\overline{k}_2 = k_1k_2k_3 + k_1k_2k_4 + k_1k_3k_4 + k_2k_3k_4 \tag{6.92}$$
$$\overline{k}_3 = k_1k_2 + k_1k_3 + k_1k_4 + k_2k_3 + k_2k_4 + k_3k_4 \tag{6.93}$$
$$\overline{k}_4 = k_1 + k_2 + k_3 + k_4 \tag{6.94}$$

Control of Lateral Position and Roll Angle Rewrite the subsystem given by (6.61)–(6.62) as $\dot{y}_1 = y_2$, $\dot{y}_2 = -g\phi_1$, $\dot{\phi}_1 = \phi_2$ and $\dot{\phi}_2 = \tilde{\tau}_{B\phi}$. Using a similar procedure to the one proposed for the pitch control, the backstepping roll control input can be obtained as

$$\tilde{\tau}_{B\phi} = \frac{\overline{k}_1}{g} y_1 + \frac{\overline{k}_2}{g} y_2 - \overline{k}_3 \phi_1 - \overline{k}_4 \phi_2 \tag{6.95}$$

6.2.3.3 Sliding Modes Control

Consider the system (6.49) of integrators in cascade, and rewrite this system in the form [52]

$$\begin{bmatrix} \dot{\boldsymbol{\varphi}} \\ \dot{\varepsilon} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_a(\varphi, \varepsilon) \\ f_b(\varphi, \varepsilon) + G(x)E(x)u + \delta(t, x, u) \end{bmatrix} \tag{6.96}$$

then, $\boldsymbol{\varphi} = [x_1\ x_2\ x_3]^{\mathrm{T}}$ and $\varepsilon = x_4$. Also $\mathbf{f}_a(\boldsymbol{\varphi}, \varepsilon) = [\alpha x_2\ \beta x_3\ \gamma x_4]^{\mathrm{T}}$ and $f_b(\boldsymbol{\varphi}, \varepsilon) = 0$, $G(x) = E(x) = 1$, $\delta(t, x, u) = 0$. Consider $x_4 = -c_0 x_1 - c_1 x_2 - c_2 x_3 = \phi(\boldsymbol{\varphi})$ in order to stabilize the origin. The partial derivative of ϕ, with respect to $\boldsymbol{\varphi}$, is given by

$$\frac{\partial \phi}{\partial \boldsymbol{\varphi}} = [-c_0 \quad -c_1 \quad -c_2] \tag{6.97}$$

then

$$\frac{\partial \phi}{\partial \boldsymbol{\varphi}} \mathbf{f}_a = -c_0 \alpha x_2 - c_1 \beta x_3 - c_2 \gamma x_4 \tag{6.98}$$

Define a sliding surface

$$s = \varepsilon - \phi(\boldsymbol{\varphi}) = x_4 + c_0 x_1 + c_1 x_2 + c_2 x_3 = 0 \tag{6.99}$$

whose time derivative is given by

$$\dot{s} = -\frac{\partial \phi}{\partial \boldsymbol{\varphi}} \mathbf{f}_a(\boldsymbol{\varphi}, \varepsilon) + u \tag{6.100}$$

then, the control input can be expressed as

$$u = -c_0 \alpha x_2 - c_1 \beta x_3 - c_2 \gamma x_4 + v \tag{6.101}$$

where $\dot{s} = v$ and $v = -\nu(x) \tanh(\frac{s}{\varepsilon})$. By definition, one has

$$\dot{s} = g(x)v + \Delta(t, x, u) \tag{6.102}$$

thus $g(x) = 1$ and $\Delta(t, x, u) = 0$. Also, one needs to satisfy

$$\left| \frac{\Delta(t, x, u)}{g(x)} \right| \leq \rho(x) + k_0 \|v\| \tag{6.103}$$

then

$$\left| \frac{0}{1} \right| \leq B \tag{6.104}$$

where B is a positive constant, thus $\rho(x) = B$ and $k_0 = 0$. One has then

$$\nu(x) \geq \frac{\rho(x)}{1 - k_0} \geq B \tag{6.105}$$

Now it can be determined that $v = -\nu(x) \tanh(\frac{s}{\varepsilon})$ or $v = -B \tanh(\frac{s}{\varepsilon})$. Note that $\tanh(s/\varepsilon)$ is a smooth approximation of the function $\mathrm{sign}(s)$, which is used in order to reduce the chattering effect. ε is selected as a small constant. Finally, it is now possible to completely write the input signal equation as

$$u = -c_0 \alpha x_2 - c_1 \beta x_3 - c_2 \gamma x_4 - B \tanh\left(\frac{s}{\varepsilon}\right) \tag{6.106}$$

For the forward-pitch subsystem equations given by (6.58)–(6.59), and by analogy with the system (6.49), one obtains $\alpha = 1$, $\beta = g$, and $\gamma = 1$, also, one has that $x_1 = x$, $x_2 = \dot{x}$, $x_3 = \theta$, and $x_4 = \dot{\theta}$. With this information, the sliding mode surface is given by

$$s = c_0 x + c_1 g \dot{x} + c_2 \theta + \dot{\theta} \tag{6.107}$$

and the forward-pitch control input can be expressed as

$$\tilde{\tau}_\theta = -c_0 \dot{x} - c_1 \theta - c_2 \dot{\theta} - B \tanh\left(\frac{s}{\varepsilon}\right) \tag{6.108}$$

The constant terms c_i should be carefully selected to obtain a stable output. In order to stabilize x in a position outside of the origin, one must place $x_1 = e_x$.

Following a similar procedure the lateral-roll control input is given by

$$\tilde{\tau}_\phi = -c_0 \dot{y} - c_1 \phi - c_2 \dot{\phi} - B \tanh\left(\frac{s}{\varepsilon}\right)$$

6.2.4 *Experimental Applications*

During this studies, three similar experiments were performed, with the purpose of identifying the most appropriate control strategy for stabilizing the position of the quad-rotor in hover flight. The experiments followed a similar procedure than the one described in Sect. 6.1.6.

Once the UAV is located exactly on top of the landing pad, the operator uses the joystick of the supervisory ground station to define the current vehicle's position as the desired $(x_\mathrm{d}, y_\mathrm{d})$ position reference. The desired altitude z_d is always fixed at 150 cm, and the desired yaw angle ψ is fixed at 0° degrees. The parameter values used for the altitude and the yaw controller are: $k_{pz} = 0.68$, $k_{vz} = 1.6$, $k_{p\psi} = 38$, $k_{p\psi} = 1350$. The parameters used for the nested saturations controller are: $\sigma_{b_4} = 0.4700$, $\sigma_{b_3} = 0.2349$, $\sigma_{b_2} = 0.1174$, and $\sigma_{b_1} = 0.0287$. The parameters used for

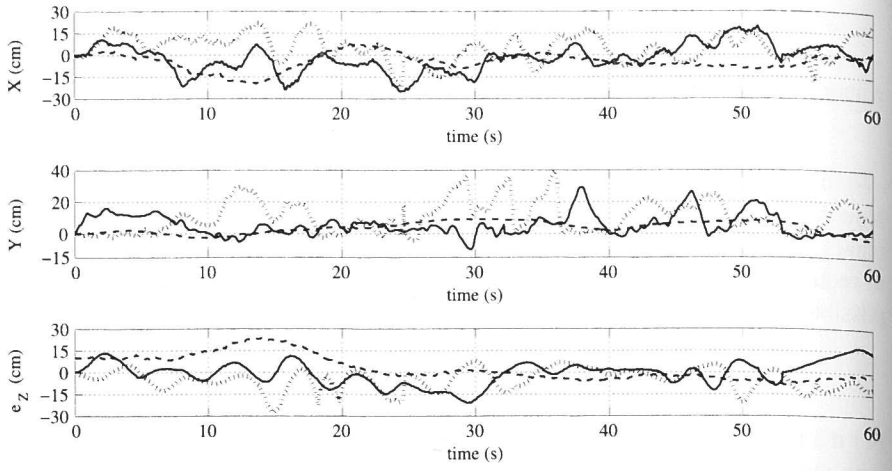


Fig. 6.9 Behavior of the x, y, and altitude error signals. The backstepping, nested saturation and sliding modes are represented by *solid line*, *dashed line*, and *dotted line*, respectively

the backstepping controller are: $\overline{k}_1 = 0.002$, $\overline{k}_2 = 0.004$, $\overline{k}_3 = 9$, and $\overline{k}_4 = 4$. The parameters used for sliding mode controller are: $c_0 = 0.3$, $c_1 = 0.25$, $c_2 = 0.15$, $B = 0.011$, and $\varepsilon = 0.05$. Those control parameters were found by trial and error.

Figures 6.9, 6.10, 6.11 and 6.12 show the obtained behavior when applying the three controllers to the UAV. It can be observed that all the controllers achieve hovering flight, however, smoother translational and angular behavior is obtained when using the nested saturations controller. One of the advantages of the nested saturation control technique is that it has a smooth behavior. Indeed, the saturation functions are not introducing jumps in the control input, and after a finite time the system will operate as a linear system. Furthermore, the nested saturation technique allows dealing first with the angular dynamics, which is the most important part of the vehicle stabilization, and once this is done we can deal with the stabilization of the translational dynamics. Figures 6.9, 6.10, 6.11 and 6.12, show that the backstepping and the sliding modes controllers induce faster changes in the vehicle's attitude, which as a consequence degrade the 3-dimensional position stabilization of the quad-rotor during the real-time experiments.

Tables 6.1 and 6.2 show the mean and standard deviation values for the position and Euler angles signals, respectively. Note that Tables 6.1 and 6.2 were computed with only one experiment for each controller, considering that the UAV is in steady state response. Mean square errors for the Euler angles were also computed, the values are shown in Table 6.3. These results show that the nested saturations controller is the method that induces less angular corrections, which can be considered as less control input generated during flight, and consequently, less energy consumption. Note in Tables 6.1 and 6.2 that the values for the backstepping controller are closer to the desired reference values. However, if the important objective concerns the energy consumption, the nested saturations controller should be considered as the best option.

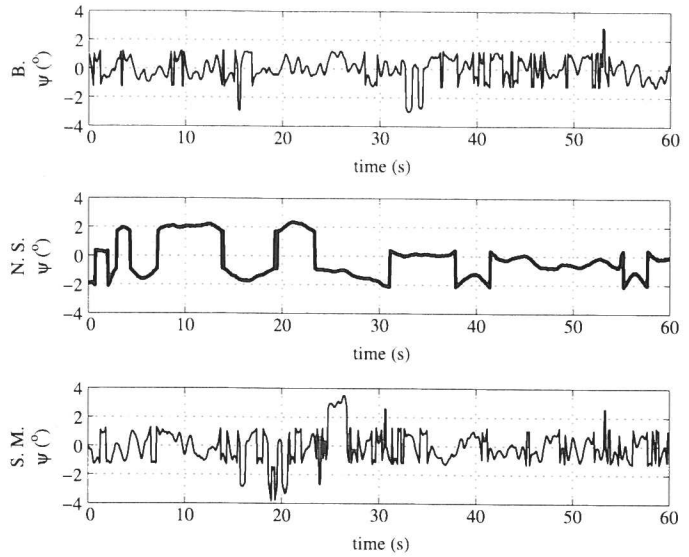


Fig. 6.10 Behavior of the yaw angle

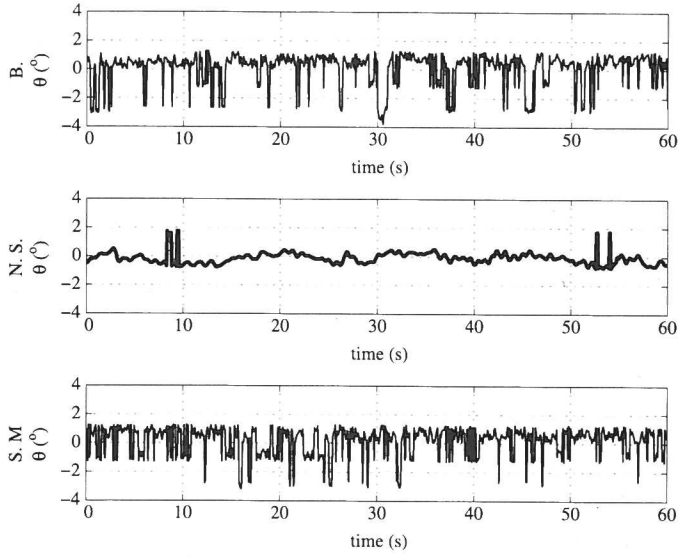


Fig. 6.11 Behavior of the pitch angle

6.2.5 *Final Comments*

From the experiments conducted during these studies, it has been observed that the three control algorithms ensure that the Euler angles of the vehicle remain very

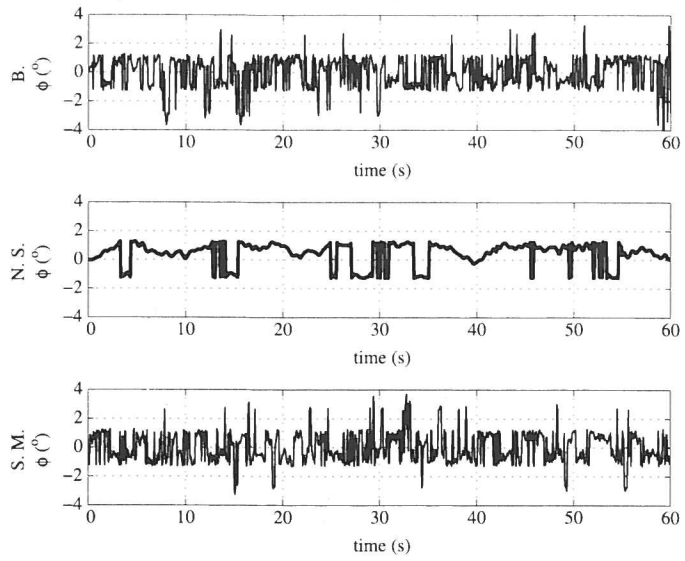


Fig. 6.12 Behavior of the roll angle

Table 6.1 Mean values of position and Euler angles

Parameter	Backstepping	Nested saturations	Sliding modes
Yaw angle	−0.0108°	−0.2504°	−0.0400°
Pitch angle	0.0903°	−0.0706°	0.2033°
Roll angle	−0.0315°	0.4371°	−0.0782°
e_Z error	0.0176 cm	1.4550 cm	−6.8905 cm
X position	−1.5662 cm	−2.7342 cm	4.3211 cm
Y position	5.6848 cm	4.2451 cm	10.1918 cm

Table 6.2 Standard deviation of position and Euler angles

Parameter	Backstepping	Nested saturations	Sliding modes
Yaw angle	0.7968°	1.2606°	1.0260°
Pitch angle	1.1588°	0.4473°	0.9833°
Roll angle	1.1673°	0.7635°	1.0739°
e_Z error	8.0868 cm	8.2162 cm	7.7071 cm
X position	9.2075 cm	6.2887 cm	10.2763 cm
Y position	7.1207 cm	4.1411 cm	9.2603 cm

close to the desired values. The obtained results show that the nested saturations control approach is the most appropriated strategy for our system, since it ensures a smoother vehicle behavior and reduces the energy consumption with respect to the other two controllers.

Table 6.3 Mean square error for Euler angles

Parameter	Backstepping	Nested saturations	Sliding modes
Yaw angle	0.6350°	1.5910°	1.0853°
Pitch	1.3519°	0.1476°	1.0144°
Roll	1.3499°	0.7623°	1.1343°

6.3 Vision-Based Altitude and Velocity Regulation

This section deals with achieving a hover flight and velocity regulation of a quad-rotor with the purpose of performing autonomous navigation. A vision system has been designed, which estimates the altitude, the lateral position and the forward speed of the helicopter during flights. It is shown that the visual information allows the construction of control strategies for different kinds of flying mode, for example hover flight and forward flight at constant speed. Experimental autonomous flights validate the visual algorithm and the control law.

6.3.1 Introduction

In this section, a monocular vision system is implemented onboard a UAV allowing altitude control, with the objective of stabilizing the position and regulating the velocity of the vehicle using optical flow. The objective of this study is to eliminate the position drift when hovering, as well as ensuring accurate displacements, two fundamental characteristics for any autonomous navigation system. If the position drift is compensated, the hover flight could be used by the system as an intermediary task between different flying modes, adapted to different conditions of the environment. Moreover, speed regulation is implemented to create the different flying modes, for example, only lateral displacement and only forward displacement. To correctly use the optical flow, a vision-based altitude controller has been also developed. The combination of these three vision-based controls allows the vehicle to navigate in a realistic application. The system is tested in two different kinds of mission: position hold over a road segment and road-following.

6.3.2 System Set-up

The overall system consists of the "Cross-Flyer" UAV platform with an embedded camera pointing downwards and the road model, as shown in Fig. 6.13. The relationship between the different reference frames can be summarized as follows:

- **Quad-rotor UAV:** evolving in space, with 6DOF. It is related to its own body coordinate system (X_B, Y_B, Z_B) and to the fixed inertial frame (X_I, Y_I, Z_I). The

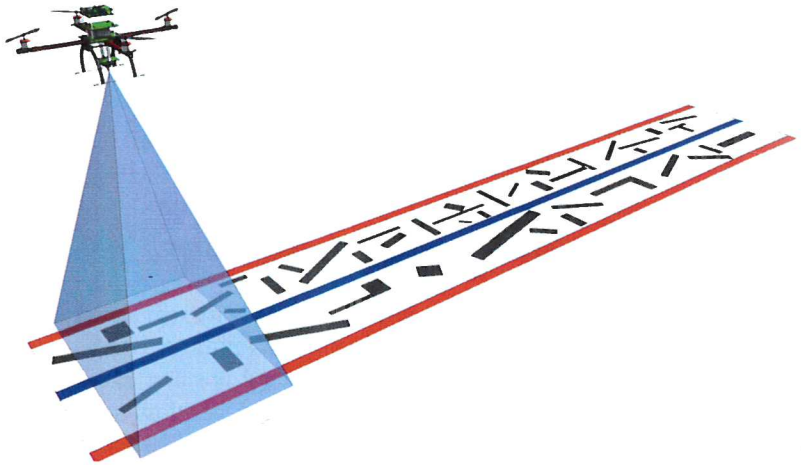


Fig. 6.13 Navigation: Visual system set-up

inertial coordinate system is located at the left corner of the beginning of the road, with the axis Z_I pointing upwards, the Y_I axis parallel to the road and the axis X_I perpendicular and coplanar with Y_I. The body coordinate system is set in the center of the rotorcraft, with Z_B the yaw axis, X_B the pitch axis and Y_B the roll axis. The planes formed by (X_B, Y_B) and (X_I, Y_I) are considered parallel since the attitude stabilization keeps the pitch and roll angles near zero.

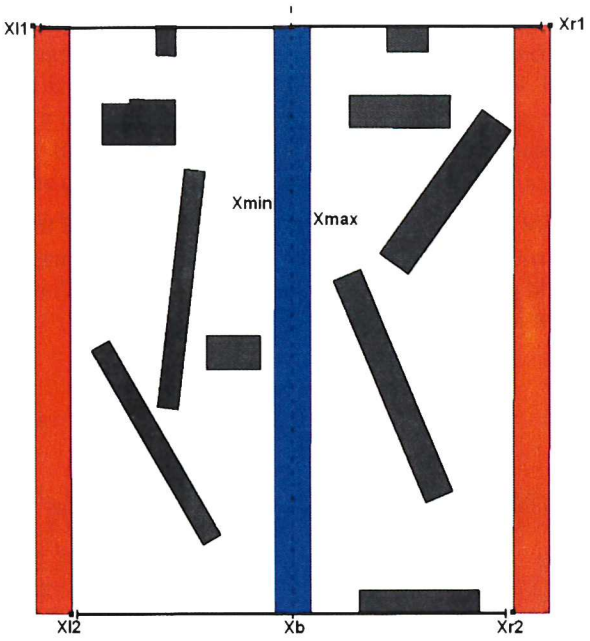
- **Pointing downwards camera:** Attached to the rotorcraft, the camera undergoes the same movement. Any sensed motion represents a displacement of the helicopter. The center of the coordinate system is located at the center of the camera, and the axis Z_C is in the opposite direction with respect to Z_I and Z_B.
- **Road model:** Represented in Figs. 6.13 and 6.14 is a dual carriageway road, with a crash barrier separating both carriageway. The crash barrier is modeled as a blue line of width K_b. Each border of the road is modeled by a red line of width K_r. The width of the road is constant and equal to W.

The imaging algorithm computes an estimate of altitude, the vehicle's position and velocities. The altitude and some position information of the engine are deduced by extracting the road zone from the image, the velocities are estimated using optical flow. All information sensed by the camera is related to the image plane, it will be shown that this information is rich enough to control the vehicle.

6.3.3 Image Processing

For a correct utilization of the optical flow, fly at a constant altitude is fundamental. To guarantee this condition, the proposed imaging algorithm computes an estimation of the altitude. Once the altitude is estimated, the vision-based velocity control can be implemented.

Fig. 6.14 The road model



6.3.3.1 Extracting the Road Zone

The first step to deduce the helicopter altitude consists of detecting the road zone, which is achieved by using a combination of line detection and color segmentation processes. The first step is to recognize the color zones by means of a classical color segmentation algorithm performed in the RGB encoding format. A threshold level τ_r between the three colors defines the red color, and a threshold τ_b defines the blue color. Once the segmentation task is done, the image is passed through a series of Gaussian filters, then, the algorithm continues with a line detection process based on a Hough transform. Finally, both results are combined to estimate the points $x^i_{l_1}$, $x^i_{l_2}$, $x^i_{\min}$, $x^i_{\max}$, $x^i_{r_1}$, and $x^i_{r_2}$. These points are image projections of the world coordinates x_{l_1}, x_{l_2}, $x_{\min}$, $x_{\max}$, x_{r_1}, and x_{r_2} representing the left border, the central separation and the right border of the road, see Fig. 6.14. With all extracted image points, the road zone is totally defined and the altitude and the position can be estimated.

6.3.3.2 Altitude and Position

The altitude estimation is done using both lines passing through the points $x^i_{l_1}$, $x^i_{l_2}$, $x^i_{r_1}$, $x^i_{r_2}$. The importance of using these points lies on having an estimate, independent of the yaw angle of the vehicle, with respect to the inertial frame (coordinates system of the road). Let us define the following variables:

$$x^i_{l_m} = \frac{x^i_{l_2} + x^i_{l_1}}{2} \tag{6.109}$$

$$x^i_{r_m} = \frac{x^i_{r_2} + x^i_{r_1}}{2} \tag{6.110}$$

$$\omega = x^i_{r_m} - x^i_{l_m} \tag{6.111}$$

where $x^i_{l_m}$, $x^i_{r_m}$ represent the border's centroids of the left and right carriageways of the road, respectively. Those points are in the image plane, and are related to the world points by the projective camera relations. Considering that all points have the same depth z, one can write

$$x^i_{l_m} = \frac{f x_l}{z} \tag{6.112}$$

$$x^i_{r_m} = \frac{f x_r}{z} \tag{6.113}$$

Introducing (6.112) and (6.113) in (6.111), one has

$$\omega = \frac{f(x_r - x_l)}{z} = \frac{fW}{z} \tag{6.114}$$

where W is the distance between the left and right boarders, i.e. the width of the road. Then

$$\frac{1}{\omega} = K_{im} z \tag{6.115}$$

where K_{im} is a constant depending on the focal length of the camera and the width of the road, and z is the altitude of the engine. From (6.115), the control of the inverse of the image-based altitude variable ω is equivalent to control the altitude z of the helicopter. The lateral position of the helicopter can also be deduced from the road extraction points. The lines passing through the points $x^i_{\min}$, $x^i_{\max}$ defines the new centroid x^i, which is the projection of the point x_b, see Fig. 6.14. The objective is to bring the image point x^i to the center of the image, which is close enough to the principal point of the image. This point can be considered as the projection of the vehicle's center of gravity, since the center of the camera and the center of the helicopter's coordinate system are aligned. Therefore, the new image-based variable can be constructed as follows:

$$\varsigma = \frac{f(x - x_b)}{z} \tag{6.116}$$

where x is the position of the helicopter's center of gravity in the inertial frame, x_b is the central line centroid, and ς is the image-based lateral position variable. A study of the time derivative is needed in order to measure the impact of an altitude variation on this variable,

$$\dot{\varsigma} = \frac{f}{z}\dot{x} - \frac{f(x - x_b)}{z^2}\dot{z} \tag{6.117}$$

$$\dot{\varsigma} = \frac{f}{z}\dot{x} + \frac{(x - x_b)}{W}\dot{\omega} \tag{6.118}$$

where $\dot{x}$ is the lateral speed of the helicopter, $\dot{\varsigma}$ is the time derivative of the image-based lateral position variable, and $\dot{\omega}$ is the time derivative of the image-based altitude variable. From (6.118) it is deduced that a previous stabilization of $\dot{\omega}$ is a necessary requirement to fulfill a good lateral position stabilization. Once the altitude is maintained constant, the time derivative $\dot{\varsigma}$ will only depend of the lateral speed of the helicopter, any other contribution will be considered as noise. Finally, when the altitude is stabilized $\dot{\omega} = 0$, then the time derivative of the image-based variables becomes

$$\dot{\varsigma} = \frac{f}{z}\dot{x} \tag{6.119}$$

6.3.3.3 Translational Velocities

Using a similar procedure as the one followed in Sect. 6.1.3.2, let us consider the camera moving with respect to a rigid scene. The velocities and rotation rates of the camera in the inertial frame are expressed by (V_x, V_y, V_z) and (w_x, w_y, w_z), respectively. To accurately estimate the pseudo-velocities of the engine, a tracking zone inside the road model is defined in a way that the point $x^i = f\frac{x_b}{z}$ is the centroid of the zone. The optical flow computed at point (x^i_k, y^i_k) is composed of a translational and a rotational part as

$$\begin{bmatrix} \mathrm{OF}_{x^i_k} \\ \mathrm{OF}_{y^i_k} \end{bmatrix} = \mathbf{T}_{\mathrm{OF}_k} + \mathbf{R}_{\mathrm{OF}_k} \tag{6.120}$$

with the translational part

$$\mathbf{T}_{\mathrm{OF}_k} = \frac{1}{z}\begin{bmatrix} -f & 0 & x^i_k \\ 0 & -f & y^i_k \end{bmatrix}\begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} \tag{6.121}$$

and the rotational part

$$\mathbf{R}_{\mathrm{OF}} = \begin{bmatrix} \frac{x^i_k y^i_k}{f} & -(f + \frac{(x^i_k)^2}{f}) & y^i_k \\ (f + \frac{(y^i_k)^2}{f}) & -\frac{x^i_k y^i_k}{f} & -x^i_k \end{bmatrix}\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{6.122}$$

The spatial mean of the optical flow (mean of the optical flow on all tracked points) is then

$$\bar{\mathrm{OF}}_x = \bar{V}_{\mathrm{OF}_x} + K^x_x \bar{V}_{\mathrm{OF}_z} + \bar{R}_{\mathrm{OF}_x} \tag{6.123}$$

$$\bar{\mathrm{OF}}_y = \bar{V}_{\mathrm{OF}_y} + K^y_y \bar{V}_{\mathrm{OF}_z} + \bar{R}_{\mathrm{OF}_y} \tag{6.124}$$

Using the results presented in [68], the rotational optical flow is compensated and the pseudo-speeds $(\bar{V}_{\mathrm{OF}_x}, \bar{V}_{\mathrm{OF}_y}, \bar{V}_{\mathrm{OF}_z})$ are deduced. Since the camera system and the helicopter share the same movements, one can write

$$\bar{V}_{\mathrm{OF}_x} = -\frac{f\dot{x}}{z} = -\dot{\varsigma} \tag{6.125}$$

$$\bar{V}_{\mathrm{OF}_y} = -\frac{f\dot{y}}{z} = -\dot{y}^i \tag{6.126}$$

$$\bar{V}_{\mathrm{OF}_z} = \frac{\dot{z}}{z} \tag{6.127}$$

where $(\dot{x}, \dot{y}, \dot{z})$ is the speed vector of the rotorcraft center of gravity. From (6.115), (6.125), and (6.126), note that the visual system allows altitude, position and velocity control of the rotorcraft using only the image-based variables $(\varsigma, \dot{\varsigma})$, $(\frac{1}{\omega}, (\frac{\dot{1}}{\omega}))$ and $\dot{y}^i = -\bar{V}_{\mathrm{OF}_y}$.

6.3.4 Control Strategy

In this study, a similar hierarchical control strategy than the one presented in Sect. 3.5 was implemented. The first two subsystems to be stabilized are the altitude and the yaw dynamics.

6.3.4.1 Altitude and Yaw Subsystems

The vision-based altitude controller is deduced from (6.11) and (6.115), then, the altitude can be stabilized with the feedback input

$$u = \frac{m(g - k_1^\omega(\frac{1}{\omega} - \frac{1}{\omega_{\mathrm{ref}}}) - k_2^\omega \frac{d}{dt}\frac{1}{\omega})}{\cos\theta\cos\phi} \tag{6.128}$$

where $\frac{1}{\omega}$ is the visual estimation of the altitude, and $\frac{d}{dt}\frac{1}{\omega}$ is computed by numerical differentiation of the variable $\frac{1}{\omega}$. For the design of this control law, the pseudo-speed in the direction of the optical axis $\bar{V}^d_{\mathrm{OF}_z}$, estimated with the optical flow algorithm, is not used. When u is implemented, the dynamic of the altitude becomes

$$\ddot{z} = \cos\theta\cos\phi\frac{u}{m} - g \tag{6.129}$$

$$\ddot{z} = -k_1^\omega\left(\frac{1}{\omega} - \frac{1}{\omega_{\mathrm{ref}}}\right) - k_2^\omega\frac{d}{dt}\frac{1}{\omega} \tag{6.130}$$

Then, by choosing the gains $k_1^\omega = K_{im}k_p^z$ and $k_2^\omega = K_{im}k_d^z$, the closed-loop dynamics can be seen as follows:

$$\ddot{z} = -k_p^z(z - z_{\mathrm{ref}}) - k_d^z\dot{z} \tag{6.131}$$

The gains k_p^z and k_d^z are chosen to ensure that the polynomial $s^2 + k_d^z s + k_p^z$ is Hurwitz. If this assumption is validated, the altitude will converge to the reference value z_{ref}. Once the altitude converges, the longitudinal and the lateral subsystems

can be seen as linear subsystems. For the yaw dynamics, one has an independent system composed of two integrators in cascade. In order to ensure the stabilization of this subsystem, the following control strategy is proposed:

$$\tilde{\tau}_\psi = -k_2^\psi\dot{\psi} - k_1^\psi\psi \tag{6.132}$$

The gains k_1^ψ and k_2^ψ are chosen appropriately so that the polynomial $s^2 + k_2^\psi + k_1^\psi$ is Hurwitz and the yaw dynamics converges to zero.

6.3.4.2 Longitudinal Subsystem

The closed-loop dynamics of θ, viewed by the navigation control, becomes

$$\dot{y}_1 = y_2^{\mathrm{ref}} + \tilde{y}_2 \tag{6.133}$$

$$\dot{\tilde{y}}_2 = \tilde{y}_3 \tag{6.134}$$

$$\dot{\tilde{y}}_3 = \tilde{\tau}_\theta \tag{6.135}$$

one chooses

$$y_2^{\mathrm{ref}} = -k_1(\bar{V}^d_{\mathrm{OF}_y} - \bar{V}^d_{\mathrm{ref}}) \tag{6.136}$$

which leads to

$$y_2^{\mathrm{ref}} = -k_p^y y_1 \tag{6.137}$$

and

$$\tilde{\tau}_\theta = -k_d^\theta\dot{\theta} - k_p^\theta(\theta - \theta_{\mathrm{ref}}) \tag{6.138}$$

where $\theta_{\mathrm{ref}} = -k_1(\bar{V}^d_{\mathrm{OF}_y} - \bar{V}^d_{\mathrm{ref}})$. The closed-loop system is represented by the polynomial $s^3 + k_d^\theta s^2 + k_p^\theta s + k_p^\theta k_p^y$, which is Hurwitz if $k_d^\theta, k_p^\theta, k_p^y > 0$ and $k_d^\theta > k_p^y$. Then, the subsystem $\dot{y} - \theta$ converges to the desired references.

6.3.4.3 Lateral Subsystem

The implementation of the vision-based control to the lateral dynamics gives

$$\dot{x}_1 = x_2 \tag{6.139}$$

$$\dot{x}_2 = x_3^{\mathrm{ref}} + \tilde{x}_3 \tag{6.140}$$

$$\dot{\tilde{x}}_3 = x_4 \tag{6.141}$$

$$\dot{x}_4 = \tilde{\tau}_\phi \tag{6.142}$$

where $x_3^{\mathrm{ref}} = \phi_{\mathrm{ref}}$. This reference angle is computed by using $(\varsigma, \dot{\varsigma})$ from (6.117) and (6.123), respectively. The reference angle dynamics are ignored since the rotational dynamics of the vehicle are faster than the translational dynamics and also because

the low-level autopilot was designed using high gains. The image-based variables $(\varsigma, \dot{\varsigma})$ are used to stabilize the position, as follows:

$$x_3^{\text{ref}} = \phi_{\text{ref}} = -k_2^x \dot{\varsigma} - k_1^x \varsigma \tag{6.143}$$

$$\phi_{\text{ref}} = -k_d^x \dot{x} - k_p^x (x - x_b) \tag{6.144}$$

Then

$$\tilde{\tau}_\phi = -k_d^\phi \dot{\phi} - k_p^\phi (\phi - \phi_{\text{ref}}) \tag{6.145}$$

The closed-loop system is stable if $k_p^\phi, k_d^\phi, k_p^x, k_d^x > 0$, $k_d^\phi > k_d^x$, and $k_p^\phi > \frac{(k_d^\phi)^2 k_p^x}{k_d^x (k_d^\phi - k_d^x)}$. Finally, the roll and the lateral displacement converge to their references.

6.3.5 Experimental Application

The experimental implementation is realized in the "Cross-Flyer" UAV platform, with the visual system detailed in Sect. 5.4.3. Two main experiments were performed: hover flight and forward flight at constant speed. The goal of such experiments is to validate the vision-based control strategy described proposed in this section.

The first experiment consists on a hover flight stabilization, over a specified zone of the road model. The hover flight is activated after a manual take-off. During experiments, the total width of the road model needs to be under the field of view of the camera in order to estimate the variable $\frac{1}{\omega}$. For this reason, the control strategy is activated once the vehicle is already flying at a desired altitude, stabilizing the rotorcraft at the current position. Figures 6.15, 6.16, and 6.17 show the behavior of the vehicle's states during real-time experiments. It can be observed in Fig. 6.17 that the Euler angle measured remains small. A video of the quad-rotor while performing such experiment can be seen at http://www.youtube.com/watch?v=xPb-IHSsNIo&feature=related.

For the second experiment, the vehicle is first set on manual mode and positioned exactly over the road model, then, the automatic speed regulation is activated via the joystick control mode. This allows the vision-based system to take the control of the quad-rotor. During tests, the vehicle sends all the information needed to analyze the experiment to the base station. The set of Figs. 6.18, 6.19, and 6.20 show the behavior of the helicopter position, velocities and Euler angles, respectively, for the velocity regulation experiment. Figure 6.19 shows how the lateral velocity is kept near zero, while the forward velocity converges to the desired value, which was set to a value of 5. This value is not the real velocity but the optical flow value in the forward direction. The altitude of the engine is also satisfactorily stabilized. It can be seen in Figs. 6.18 and 6.19 that the altitude z and velocity $\dot{z}$ present only small changes. A video of the UAV flying forward at constant speed can be seen at http://www.youtube.com/watch?v=PpUW9a3S3GQ&feature=related.

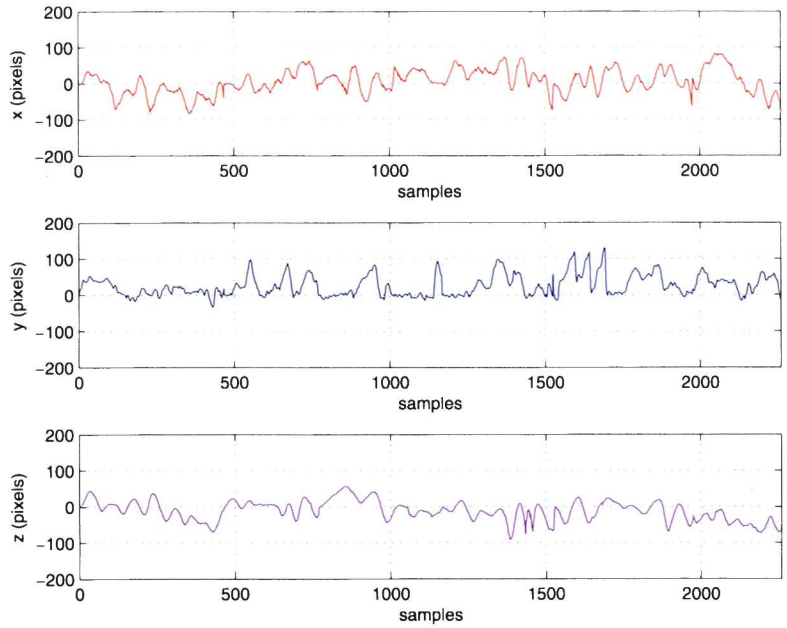


Fig. 6.15 Hover stabilization: Position results

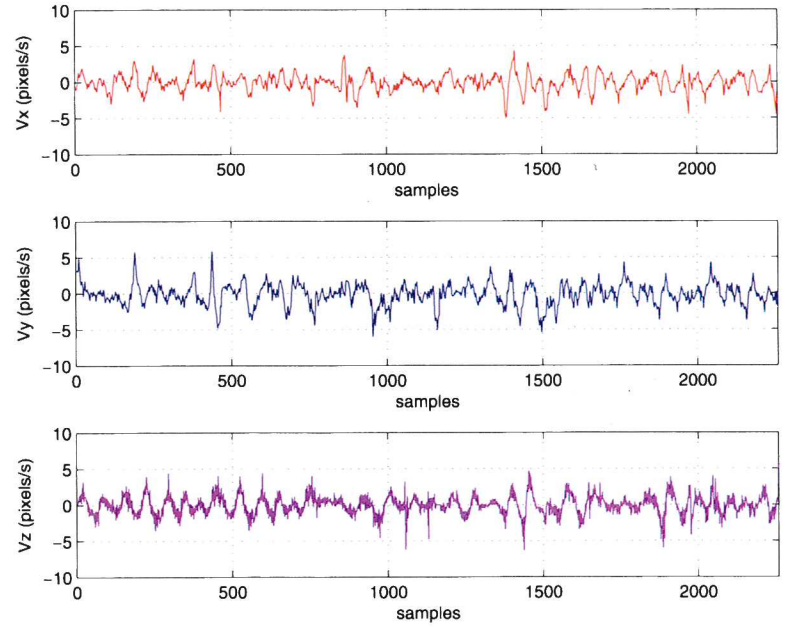


Fig. 6.16 Hover stabilization: Velocities results

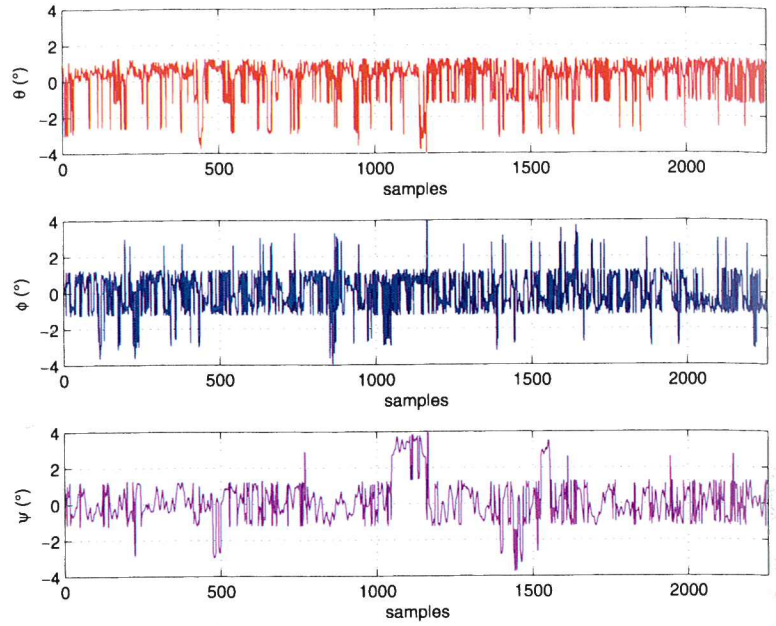


Fig. 6.17 Hover stabilization: Euler angles results

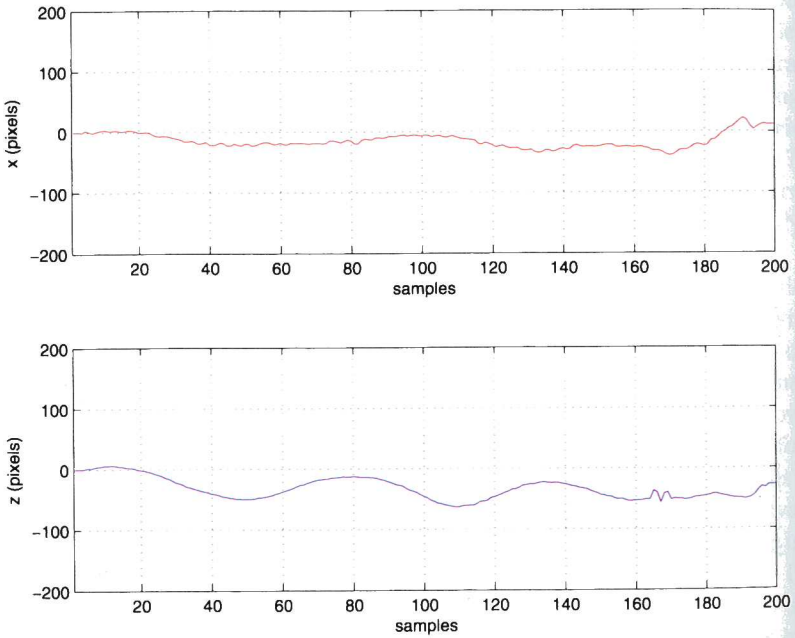


Fig. 6.18 Velocity regulation: Position results

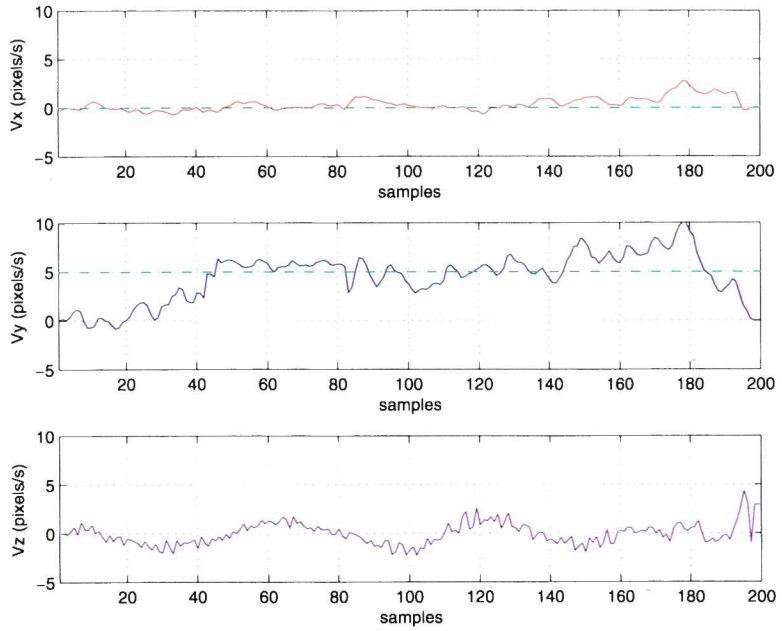


Fig. 6.19 Velocity regulation: Velocities results

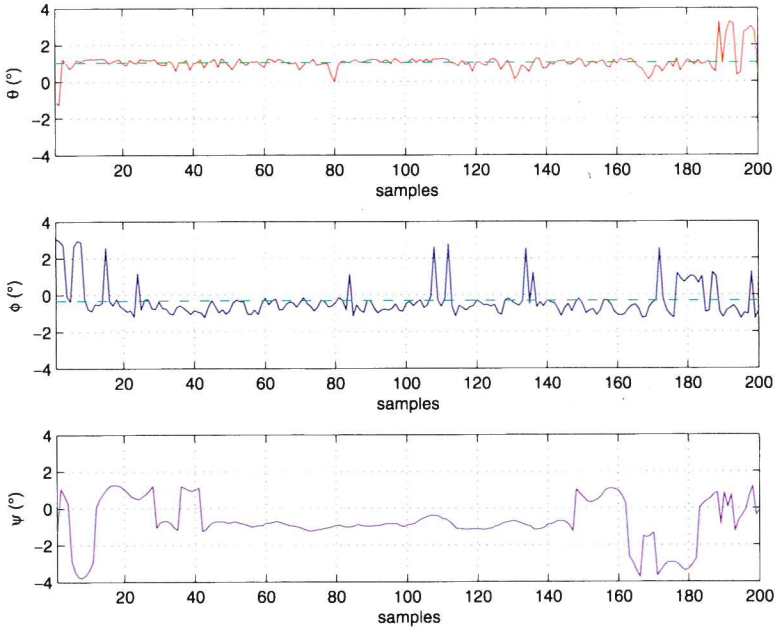


Fig. 6.20 Velocity regulation: Euler angles results

6.3.6 *Final Comments*

In this section, a vision-based strategy for 3-dimensional position control and velocity regulation was proposed and tested in real-time applications. Using the UAV quad-rotor platform, the proposed controller and vision algorithm were tested on a mission consisting on road-following. The estimated information proved to be rich enough to allow performing navigation missions. It has been shown that the vision-based controller does not cause instability in the vehicle attitude.

6.4 Concluding Remarks

In this chapter, two different vision-based strategies for stabilizing a quad-rotor UAV during flight were proposed and tested in real-time applications.

The first strategy consists of an homography estimation technique for extracting the 3-dimensional position, as well as of an optical flow computation for deriving the vehicle's translational velocities. A technique for predicting the landing pad position when its detection fails was presented also. The experimental results validate the effectiveness of such approach. In a second phase, three different control strategies were implemented over the same experimental set-up, with the purpose of identifying the most effective controller for stabilizing the vehicle when using visual feedback. The second vision-based strategy was developed for controlling the helicopter's 3-dimensional position, as well as achieving a velocity regulation. The proposed controller and the vision algorithm were tested on a mission consisting on road-following. The estimated information proved to be rich enough to allow performing navigation missions. It has been shown that the vision-based controller does not cause instability in the vehicle attitude.

From the set of experiments performed, it was found that the most difficult task corresponds to the altitude stabilization. Indeed, the delay between the dynamics and the visual estimation could cause the system to oscillate. Moreover, since the battery is continuously consuming when the rotorcraft flies, the system power decreases while the time passes. Thus, the rotorcraft tends to fall more easily than it rises. This problem can be solved by making the altitude control asymmetric, providing more importance to the positive control, which deals with the ascension, over the negative control which allows the descent of the rotorcraft. With this arrangement, the negative control is saturated faster, causing the rotorcraft to descend slowly when it overruns the desired altitude.

A major disadvantage of both vision-based strategies is that, for a proper functioning, the artificial landmarks must be completely detected by the imaging sensor. For the first strategy, this requirement must be fulfilled keeping the landing pad under the field of view of the camera. Concerning the second approach, the total width of the road model needs to be under the field of view of the camera, in order to estimate the variable $\frac{1}{\omega}$.

Aiming at overcoming the issues generated by the limitations of the monocular imaging sensors, a stereo vision system could be implemented, since it allows to directly estimate the relative 3-dimensional position of the imaging system with respect to its surrounding environment. The next chapter is devoted to the implementation of a stereo imaging system for the quad-rotor.

Chapter 7
Combining Stereo Imaging, Inertial and Altitude Sensing Systems for the Quad-Rotor

From previous chapters it is evident that, although monocular systems are low-weight designs and allow performing in an acceptable manner several vision-based tasks, they suffer from some drawbacks. One major inconvenience is that, in order to estimate its relative 3-dimensional position, they require using artificial markers of known dimensions. In addition, those markers must be inside the camera's field of view in every instant of time, which is commonly very hard to accomplish. A very promising solution to overcome those issues consists of the implementation of stereo vision systems and complementary sensors, which allow using the surrounding natural environment in order to estimate relative position.

In this chapter, a quad-rotor robotic platform equipped with a stereo imaging, inertial and altitude sensing system is presented. The objective of this research consists of enabling the UAV to autonomously perform take-off, relative positioning, navigation and landing, when evolving in unstructured, indoors, and GPS-denied environments.

The chapter is divided as follows. Section 7.1 deals with the problem of estimating the relative motion of a quad-rotor UAV in all six degrees of freedom. Aiming at this objective, an imaging, inertial, and altitude sensing system is introduced, which allows computing the 3-dimensional position and translational velocity of the vehicle with respect to its surrounding environment. The techniques for estimating the vehicle ego-motion and performing sensors fusion are presented in detail. Some experimental results consisting of motion estimation show the effectiveness of the proposed approach. With the purpose of identifying the most effective approach for combining visual odometry with inertial measurements, a Luenberger observer, a Kalman filter and a complementary filter are studied and compared in Sect. 7.2. To evaluate the performance of each strategy, real-time experiments consisting of motion detection and autonomous relative positioning are shown. Finally, some concluding remarks are presented in Sect. 7.3.

7.1 Estimating Motion

This Section presents the development of an imaging, inertial and altitude sensing system for the "Improved X-Flyer" quad-rotor. By using the images provided by a

L.R. García Carrillo et al., *Quad Rotorcraft Control*, Advances in Industrial Control,
DOI 10.1007/978-1-4471-4399-4_7, © Springer-Verlag London 2013

pair of cameras, a visual odometry algorithm allows computing the ego-motion of the vehicle with respect to its surrounding environment. With the purpose of estimating the vehicle's 3-dimensional position and translational velocity in an accurate way, vision-based position estimation is combined with IMU-based accelerations and ultrasonic-based altitude measurements in a state estimator strategy.

7.1.1 Introduction

A basic requirement for UAVs consists of robust autonomous positioning and navigation. Information concerning the angular behavior of the aircraft are commonly estimated by means of inertial sensors (gyro, accelerometer, IMU), while the most common approach for estimating the 3-dimensional position and translational velocity is based on a GPS, which can be used in most of outdoor environments. In recent years, interest on the development of UAVs that can operate indoor or in urban environments has increased [41, 46, 84], thereby enabling an even wider range of robotic tasks to be accomplished. Unfortunately, when UAVs are required to navigate through urban environments or indoors, GPS signals may be noisy or even unavailable, leading to a poor or undesirable performance of the vehicle. Previous situations have motivated the search for alternative sensing solutions.

Fusion of data provided by a group of complementary sensors has proved to be an appropriate approach for fully estimating the state variables of a vehicle, as well as for sensing its surrounding environment. However, when working with a mini-UAV like the quad-rotor helicopter, the group of sensors and embedded electronics should be carefully chosen due to payload limitations. Taking into account the limitations just mentioned, the "Improved X-Flyer" robotic helicopter presented in Sect. 3.4.3 has been equipped with an IMU that provides the vehicle's attitude, as well as with three analog rate sensors to measure the platform angular rate. To estimate the helicopter's altitude at low and high flights, an ultrasonic sensor and an atmospheric pressure sensor are also installed. Attitude and position signals are read by an embedded micro-processor, which computes the control input that stabilizes the vehicle during flight. This basic sensor suit has been enhanced by including a stereo vision system in combination with a secondary IMU, with the purpose of exploiting complementary characteristics of imaging and inertial sensors.

Fast changes in angular rotation rates and linear accelerations are accurately measured by inertial sensors. Unfortunately, integration of these signals leads to unbounded low-frequency drift, making autonomous positioning and navigation a complex task. On the other hand, vision-based motion estimation provides good accuracy when the camera's field of view changes relatively slowly. In addition to the inertial and imaging sensors, an ultrasonic range finder is used to provide an estimation of the helicopter altitude. Performing a fusion of the three sensors data, each output information can be used to compensate for the weaknesses inherent in the others. For these studies, the helicopter motion is estimated by using visual odometry, which consists of tracking triangulated natural landmarks across sequential

Fig. 7.1 The quad-rotor robotic platform equipped with a stereo imaging, inertial, and altitude sensing system

stereo image pairs. Vision-based measurements are then combined with inertial and altitude sensor signals in a state observer, in order to provide accurate estimates of the UAV's (x, y, z) positions and $(\dot{x}, \dot{y}, \dot{z})$ translational velocities. The "Improved X-Flyer" aerial vehicle equipped with the sensing system just described is shown in Fig. 7.1.

7.1.1.1 Related Work

The process of incrementally estimating changes in robot pose by identifying and tracking visual landmarks in the environment is known as visual odometry (VO). Robotic vehicles are equipped with vision systems consisting of monocular [24], omnidirectional [31] and spherical cameras [55] in order to perform VO. An interesting result concerning vision-based localization of a UAV can be found in [13]. The authors implement a simultaneous localization and mapping algorithm (SLAM), using a monocular vision system. Their experimental results show accurate localization and positioning of a quad-rotor performing indoors. Stereo vision systems are also a common solution for estimating VO, since the depth of the tracked landmarks can be computed directly from known camera geometry. An example of systems equipped with VO based on stereo vision are the NASA Mars Exploration Rover (MER) robots [29]. During periods when wheel odometry is unreliable, such as when driving over high-slip terrain, the rovers position estimation rely only on visual estimates.

An example of combined visual and inertial sensing systems applied to aerial vehicles can be found in [30], where a vision and inertial sensing system is used for real-time control of a small helicopter. Height above the ground and optical flow due to ego-motion are provided by a pair of downward looking cameras. A complementary filter is used to fuse inertially and visually derived velocity information. In [51] the authors present a UAV navigation system which combines stereo VO with inertial measurements from an IMU. Vehicle position and attitude are obtained by fusing the motion estimates from both sensors in an extended Kalman filter. Information provided by the navigation system is analyzed off-line to evaluate the performance of a point to point navigation task.

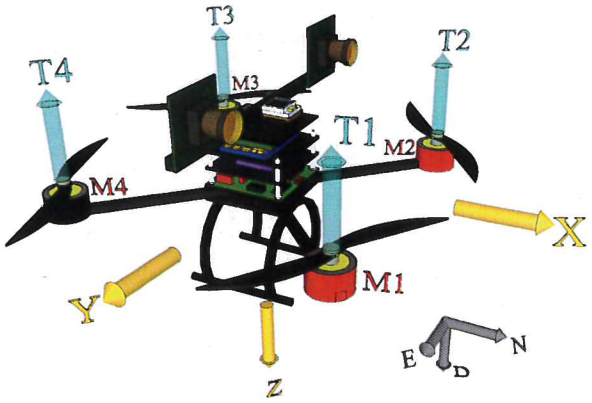


Fig. 7.2 NED diagram of the quad-rotor dynamical model

Unlike research work where experiments are performed over robotic platforms capable of carrying a considerable amount of payload (typically up to 5 kg), the present study focuses its efforts on the development of a small quad-rotor robotic helicopter equipped with a well suited sensing system. The research work presented here is closely related to aerial simultaneous localization and mapping (see, for example, [2]), although it is primarily concerned with the tasks of autonomous take-off, landing, positioning and point-to-point navigation, and so does not maintain a map of landmark positions.

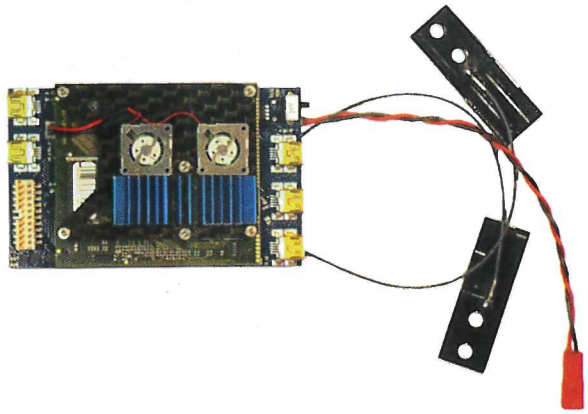
7.1.2 System Setup

The dynamic model equations for an "X-Flyer" quad-rotor having two frontal motors and two rear motors is considered, see Fig. 7.2. Such model has been previously presented in Sect. 2.2.4, (2.61)–(2.66). The "X-Flyer" quad-rotor was considered for installing the proposed sensing system because under this arrangement the motors are positioned at 45° from the X and Y axes. When using a vision system pointing forwards, which is the case, it is desirable to have a clear view of the scene in front of the helicopter.

7.1.3 Experimental Platform Overview

Small quad-rotor UAVs have a maximum amount of vertical thrust that they can generate to remain airborne, which severely limits the amount of payload available for sensing and computation. This payload limitation restricts popular choices of embedded computers such as the PC-104, large-aperture cameras and high-fidelity

Fig. 7.3 Ascending Technologies® Flying Netbook mini-computer



IMUs. Instead, small aerial robots rely on ultrasonic sensors, lightweight but lower-quality IMUs, and micro cameras having limited ranges and fields-of-view. It is common that the measurements provided by small sensors are noisier compared to their equivalents found in bigger robotics platforms.

The "Improved X-Flyer" quad-rotor helicopter is composed by a group of off-the-shelf components. Details concerning the components and electronics onboard were previously presented in Sect. 3.4.3. The robotic helicopter, as shown in Fig. 3.8, has a total weight of 945 grams. It has a payload capacity of 600 grams, which is enough for carrying the proposed sensing system described in the next section.

7.1.3.1 Visual and Inertial Navigation System

With the purpose of performing visual and inertial navigation studies, two monochromatic uEye cameras from IDS with a resolution of 376 × 240 pixels and a field of view of 75° were chosen. The cameras are mounted on a stereo bench with a 35 cm baseline and are placed facing forward. An additional 3DM-GX3 IMU from Microstrain® is mounted on the central section of the stereo bench with the purpose of measuring the acceleration of the vehicle. The cameras and the IMU are connected to an embedded Flying Netbook board developed by Ascending Technologies®, see Fig. 7.3. This mini-computer runs a program that deals with the next group of tasks. First, it synchronizes the uEye cameras, reads their images and compresses them in jpeg format. Secondly, it communicates with the 3DM-GX3 IMU and reads the acceleration data. The program generates a data buffer containing both images and acceleration values, and finally, this buffer is sent via 801.11n wireless communication to the supervisory ground station PC. Once the buffer is received, the images are decompressed and the acceleration values are read. Next, the images are processed by a VO estimation application, which runs in the 2.4 GHz ground station PC. Finally, VO estimation is combined with acceleration and altitude measurements in a state observer to provide accurate measurements of the

Fig. 7.4 Visual and inertial navigation system

helicopter's (x, y, z) positions and $(\dot{x}, \dot{y}, \dot{z})$ translational velocities, which are required data for the control algorithm. The visual and inertial system can be seen in Fig. 7.4, and mounted at the top of the "Improved X-Flyer" quad-rotor helicopter in Fig. 7.1. The portable design of this system allows an easy installation–removal procedure from the robotic helicopter, without compromising the proper functioning of the platform.

7.1.3.2 Supervisory Ground Station

A supervisory ground station well suited for receiving sensor signals and executing imaging and control applications was developed. This system consists of a computer where a supervisory control application and an imaging processing application are executed. The computer has connected a flight simulator joystick, a bidirectional wireless link for data transmission and a 801.11n wireless link for receiving data sent by the Flying netbook. The wireless data link is used to receive and save information sent periodically by the aerial vehicle, which helps to characterize its performance during experiments, but also for sending a buffer containing estimated states and control commands to stabilize the helicopter's position. The control commands can be sent either by a user (by means of a flight joystick) or generated by the supervisory control application. The 801.11n wireless link receives real-time stereo images and acceleration measurements, which are processed by a computer vision program. The position and acceleration information extracted by this algorithm is placed on a fixed memory segment that is shared with the supervisory control application, where a state observer estimates the vehicles states required in the control algorithm.

The main function of the base station is the supervisory control application, it coordinates the running programs and allows to piloting the quad-rotor manually if needed. Data transmission between the supervisory ground station and the aerial vehicle is performed at a frequency of 13 Hz. Data sent from the base station to the UAV are prioritized since they carry information necessary for position control. The supervisory ground station is shown in Fig. 7.5. Figure 7.6 shows a schematic summarizing the steps performed by the Flying Netbook to obtain the images and acceleration values, it also shows the operations carried out in the supervisory ground station for obtaining the vehicle's states and sending the buffer with the required data for the controller.

Fig. 7.5 The supervisory ground station. From left to right: joystick, PC, 801.11n wireless link, XBEE09P data link

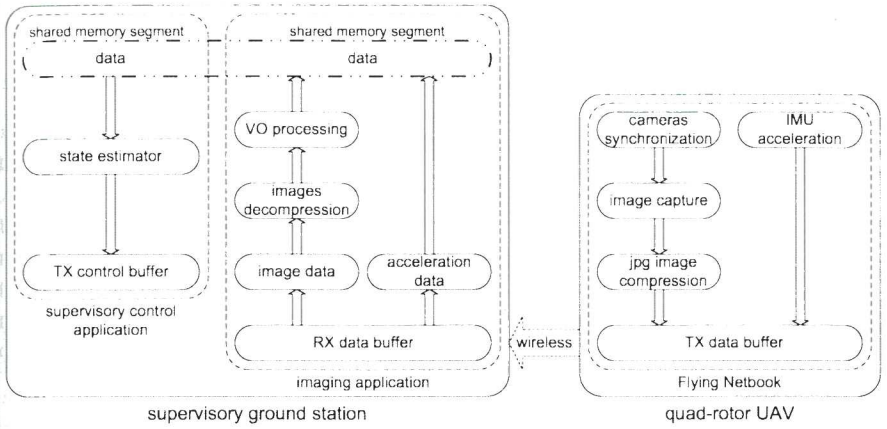


Fig. 7.6 A schematic of the process for estimating the vehicle's states

7.1.4 Stereo Visual Odometry

Stereo odometry is a vision-based technique that computes the ego-motion of a stereo rig through its surrounding environment by evaluating the camera's images. It can be thought of as a chain of several single subprocesses, where each of them relies on its predecessor's results. For each of these subprocesses, a variety of exchangeable methods are available [76].

The approach for performing stereo visual odometry is represented in Fig. 7.7. The algorithm begins in the previous left image, where features that can be tracked reliably are detected and identified. For each selected left image feature, its corresponding feature is searched in the previous right image. The 3-dimensional positions of the matched features pairs are reconstructed using triangulation. When a newer stereo images pair is available, features that have been reconstructed successfully are tracked from the previous left image to the current left image. Then, a similar correspondence and reconstruction step is performed for the current left image and current right image. At each time step, the process just described yields

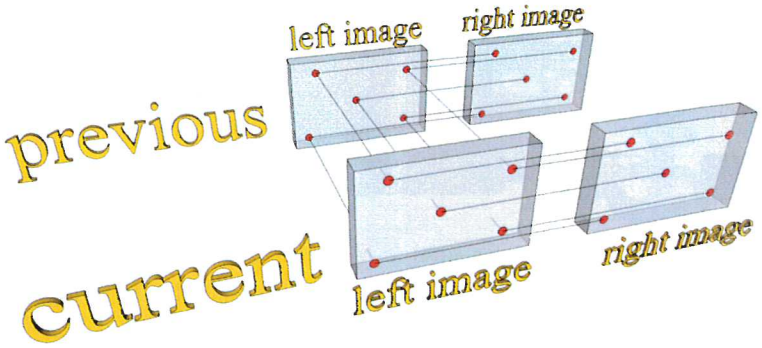


Fig. 7.7 Principle of the visual odometry algorithm

two sets of corresponding 3-dimensional features observations, before and after the helicopter platform has undergone an unknown rotation R and translation $\mathbf{T}$. Using the matched sets of features, the vehicle's relative motion in all six degrees of freedom can be computed.

7.1.4.1 Detecting Features

Finding interest features in the images is the first step towards stereo odometry. The algorithm proposed in [74] is being used to find salient characteristics in the left stereo image. In order to achieve a uniform detection of features over the entire image, four similar searching regions are defined (upper left, upper right, lower left, lower right). In addition, the parameters of the feature detector are adjusted to avoiding detection of features under some geometric proximity. The algorithm searches for a number of 150 candidate features over each searching region.

The pyramidal implementation of the Lucas–Kanade optical flow tracker [18] available in OpenCV [64] is used to track features between the left and right images, as well as from the previous to the current left frames. This algorithm allows tracking features robustly over large baselines, in addition, it is robust to the presence of image blur caused by motion. In order to validate feature correspondences between the left and right images, an error checking is performed at this stage. Based on the method available in [17], an accurate calibration of the stereo cameras has been performed and the obtained results were previously presented in Sect. 5.2.2. The calibration procedure allows the evaluation of the epipolar constraint

$$\mathbf{x}_r^{\mathrm{T}} F \mathbf{x}_l = 0 \pm S \tag{7.1}$$

which is a variation of (5.23) with an extra parameter S. In this formulation, $\mathbf{x}_*$ denotes the feature location in the respective (left or right) frame, F represents the fundamental matrix obtained from the extrinsic calibration of the stereo rig, and S is a threshold previously defined for acceptable noise level. It has been found experimentally that the vision algorithm has a good performance concerning

speed/accuracy if the number of successfully tracked features is maintained over 150. Aiming at maintaining this minimum number of successfully tracked features for every instant of time, the algorithm to find salient characteristics is executed each time the number of features is under 150.

7.1.4.2 3-dimensional Reconstruction

As soon as a pair of 2-dimensional features corresponding to a physical feature in space is found by the algorithm explained earlier, the position of this feature in 3-dimensional space can be reconstructed. A 3-dimensional point in space is projected into the left and right camera's image plane up to a scalar factor λ by [60]

$$\widehat{\mathbf{x}}_{il} = K_l \cdot \Pi_0 \cdot {}^{o}T_l \cdot \mathbf{X}_0 = \Pi_l \mathbf{X}_i \tag{7.2}$$

$$\widehat{\mathbf{x}}_{ir} = K_r \cdot \Pi_0 \cdot {}^{o}T_l \cdot {}^{l}T_r \cdot \mathbf{X}_0 = \Pi_r \mathbf{X}_i \tag{7.3}$$

where $\widehat{\mathbf{x}}_{il} = \lambda \mathbf{x}_{il}$ and $\widehat{\mathbf{x}}_{ir} = \lambda \mathbf{x}_{ir} \in \mathbb{R}^{3\times 1}$ represent the projection of $\mathbf{X}_i \in \mathbb{R}^{4\times 1}$ into the image plane of the left and right cameras through the projection matrices Π_l, $\Pi_r \in \mathbb{R}^{3\times 4}$. Each one of these matrices consist of the intrinsic parameter matrix K_*, the standard projection matrix Π_0, as well as of the extrinsic parameters matrix ${}^{l}T_r$ representing the transformation of the right camera with respect to the left camera (obtained from the extrinsic stereo calibration presented in Sect. 5.2.2), and ${}^{o}T_l$ representing the transformation from the left camera with respect to the vehicle's center of gravity, which is simply expressed as

$${}^{o}T_l = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -17.5 \\ 0 & 0 & 1 & -10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.4}$$

From (7.2) and (7.3) a homogeneous system of three equations per camera is obtained. Only two of the three equations are linear independent since $\mathrm{rank}(\widehat{x}) = 2$. This leads to four constraints for the point $\mathbf{X}_i$ that must be recovered from both views:

$$(x_l \boldsymbol{\pi}_l^{3\mathrm{T}} - \boldsymbol{\pi}_l^{1\mathrm{T}})\mathbf{X}_i = 0 \tag{7.5}$$

$$(y_l \boldsymbol{\pi}_l^{3\mathrm{T}} - \boldsymbol{\pi}_l^{2\mathrm{T}})\mathbf{X}_i = 0 \tag{7.6}$$

$$(x_r \boldsymbol{\pi}_r^{3\mathrm{T}} - \boldsymbol{\pi}_r^{1\mathrm{T}})\mathbf{X}_i = 0 \tag{7.7}$$

$$(y_r \boldsymbol{\pi}_r^{3\mathrm{T}} - \boldsymbol{\pi}_r^{2\mathrm{T}})\mathbf{X}_i = 0 \tag{7.8}$$

where x_* and y_* denote the image coordinates of the feature and $\boldsymbol{\pi}_*^{j\mathrm{T}}$ denotes the jth row vector of the projection matrix $\Pi_* = [\boldsymbol{\pi}^1\ \boldsymbol{\pi}^2\ \boldsymbol{\pi}^3]^{\mathrm{T}}$. The set of equations (7.5)–(7.8) form a 4×4 homogeneous system that must be solved:

$$M\mathbf{X}_i = \mathbf{0} \tag{7.9}$$

In practice the right-hand side of the above equation is different from zero due to inaccuracies and noise when finding correspondences or in the process of camera calibration. In other words, there is no intersection between the lines passing through each optical center of the cameras and $\mathbf{x}_l - \mathbf{x}_r$. Computing the eigenvalue decomposition of $M^{\mathrm{T}}M$ a solution that minimizes the error $\|M\mathbf{X}_i\|$ of the homogeneous system in (7.9) in a least squares sense is obtained. The best solution for $\mathbf{X}_i$ is the eigenvector of $M^{\mathrm{T}}M$ corresponding to the smallest eigenvalue. In order to complete the 3-dimensional reconstruction, $\mathbf{X}_i$ is normalized to make its last coordinate (X_4) equal to 1.

7.1.4.3 Estimating Motion

The procedure presented here follows the notation from [1]. At each time step, the reconstruction algorithm described above yields two sets of corresponding 3-dimensional features observations, before and after the helicopter stereo rig has undergone an unknown rotation R and translation $\mathbf{T}$. A method for computing the transformation between two sets of points is presented in [79]. This algorithm is intended to determine rotation R and translation $\mathbf{T}$ from the previous ($\mathbf{X}_p$) and the current ($\mathbf{X}_c$) set of points that minimize the mean square errors $e^2(R, \mathbf{T})$. The transformation can be described as

$$\mathbf{X}_{c,i} = R \cdot \mathbf{X}_{p,i} + \mathbf{T} \tag{7.10}$$

then

$$e^2(R, \mathbf{T}) = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{X}_{c,i} - R \cdot \mathbf{X}_{p,i} - \mathbf{T}\|_2^2 \tag{7.11}$$

where $i = 1, \ldots, n$ denotes the number of features, and there must be at least three of them in order to uniquely estimate the transformation. First the rotation must be estimated, therefore the set of points are translated by their mean vectors $\overline{\mathbf{X}}_p$ and $\overline{\mathbf{X}}_c$ to the origin:

$$\overline{\mathbf{X}}_p = \frac{1}{n} \sum_{i=1}^{n} \mathbf{X}_{p,i} \quad \tilde{\mathbf{X}}_{p,i} = \mathbf{X}_{p,i} - \overline{\mathbf{X}}_p \tag{7.12}$$

$$\overline{\mathbf{X}}_c = \frac{1}{n} \sum_{i=1}^{n} \mathbf{X}_{c,i} \quad \tilde{\mathbf{X}}_{c,i} = \mathbf{X}_{c,i} - \overline{\mathbf{X}}_c \tag{7.13}$$

Using the translated set of points $\tilde{\mathbf{X}}_{p,i}$ and $\tilde{\mathbf{X}}_{c,i}$ a 3×3 covariance matrix can be computed as

$$\Sigma_{pc} = \frac{1}{n} \sum_{i=1}^{n} \tilde{\mathbf{X}}_{c,i} \cdot \tilde{\mathbf{X}}_{p,i}^{\mathrm{T}} \tag{7.14}$$

R is obtained by performing the singular value decomposition (SVD) $\Sigma_{pc} = USV^{\mathrm{T}}$. Finally, to obtain as result a rotation and not a reflection $\det(U) \cdot \det(V)$ must be evaluated. If necessary, S must be corrected as

$$R = U\tilde{S}V^{\mathrm{T}} \tag{7.15}$$

$$\tilde{S} = \begin{cases} \mathrm{diag}(1, 1, 1) & \text{for } \det(U) \cdot \det(V) \geq 0 \\ \mathrm{diag}(1, 1, -1) & \text{for } \det(U) \cdot \det(V) < 0 \end{cases} \tag{7.16}$$

Once rotation is known, translation can be computed by inserting the two mean vectors $\overline{\mathbf{X}}_p$ and $\overline{\mathbf{X}}_c$ into (7.10):

$$\mathbf{T} = \overline{\mathbf{X}}_c - R\overline{\mathbf{X}}_p \tag{7.17}$$

which represents the transformation of two sets of points with respect to (w.r.t.) a fixed coordinate system. In the present case, the set of points is fixed while the stereo cameras (helicopter) coordinate system is moving. The rotation and translation of the helicopter to its previous body frame are obtained by

$$\Delta R = R^{\mathrm{T}} \tag{7.18}$$

$$\Delta \mathbf{T} = -R^{\mathrm{T}}\mathbf{T} \tag{7.19}$$

The Δ factor represents that the pose of the vehicle is only computed in the current time step w.r.t. the previous one. In order to estimate the pose of the vehicle w.r.t. its starting pose T_0, homogeneous transformations must be applied to perform a chaining of all of the Δ poses obtained:

$$T_{\text{current}} = \underbrace{T_0 \cdot \Delta T_{t-m+1} \cdot \ldots \cdot \Delta T_{t-1}}_{T_{\text{previous}}} \cdot \Delta T_t \tag{7.20}$$

with

$$T = \begin{bmatrix} R & \mathbf{T} \\ \underline{0} & 1 \end{bmatrix} \tag{7.21}$$

By performing a right-multiplication of the previous pose T_{previous} with the latest ΔT_t pose, the current motion w.r.t. the previous pose is added to the previous estimated pose. The pose where the vehicle started is represented by T_0, and, in the present case, it corresponds to the identity matrix. The group of last Δ poses can be stored in the single 4 × 4 matrix T_{previous}. Therefore, the pose update in every time step requires only one matrix multiplication. In order to describe the helicopter's attitude from VO, the (ψ, θ, ϕ) Euler angles must be extracted from the rotation matrix R as

$$\theta = \mathrm{atan2}\left(-r_{31}, \sqrt{r_{11}^2 + r_{12}^2}\right) \tag{7.22}$$

$$\psi = \mathrm{atan2}\left(\frac{r_{21}}{\cos(\theta)}, \frac{r_{11}}{\cos(\theta)}\right) \tag{7.23}$$

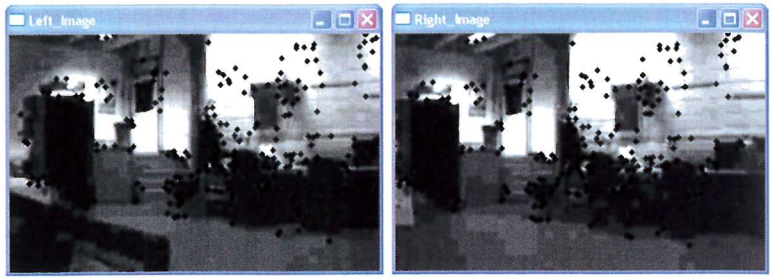


Fig. 7.8 The visual odometry algorithm running on the stereo images

$$\phi = \text{atan2}\left(\frac{r_{32}}{\cos(\theta)}, \frac{r_{33}}{\cos(\theta)}\right) \tag{7.24}$$

where $r_{\text{row,column}}$ denotes a specific entry of R.

The least square method explained cannot be used without any further improvement to make it robust against outliers in the data. RanSaC [35] is a well known method for robust estimation. As in [62], it has been decided to implement the SVD algorithm as an hypothesis generator for the RanSaC procedure. Once the hypothesis with the maximum number of inliers is found, the solution is recomputed using all the inliers. An image of the visual odometry algorithm running on the stereo images is presented in Fig. 7.8. The scene corresponds to an unstructured indoors environment. The tracked features used for obtaining visual odometry are highlighted with black dots.

7.1.5 A Simple Strategy for Imaging, Inertial and Altitude Data Fusion

With the purpose of obtaining an estimate of the vehicle's 3-dimensional position and velocity, imaging, inertial and altitude measurements are fused in a Kalman filter. The state vector is defined as

$$\mathbf{x}[k] = [\, x \quad \dot{x} \quad \ddot{x} \quad y \quad \dot{y} \quad \ddot{y} \quad z \quad \dot{z} \quad \ddot{z} \,]^{\mathrm{T}} \tag{7.25}$$

with (x, y, z) representing the position of the quad-rotor in the global NED frame. The observation vector is defined as

$$\mathbf{z}[k] = [\, x_{\text{vo}} \quad x_{a,\text{imu}} \quad y_{\text{vo}} \quad y_{a,\text{imu}} \quad z_{\text{vo}} \quad z_{a,\text{imu}} \quad z_{\text{as}} \,]^{\mathrm{T}} \tag{7.26}$$

where $(x_{\text{vo}}, y_{\text{vo}}, z_{\text{vo}})$ represents measurements of the helicopter position provided by the visual odometry algorithm, and $(x_{a,\text{imu}}, y_{a,\text{imu}}, z_{a,\text{imu}})$ are linear accelerations provided by the IMU. The measurement provided by the altitude sensor is represented by z_{as}. The filter fuses z_{vo} and z_{as} in order to compute a better estimation of the vehicle's altitude. The Kalman filter functions used in this experiment

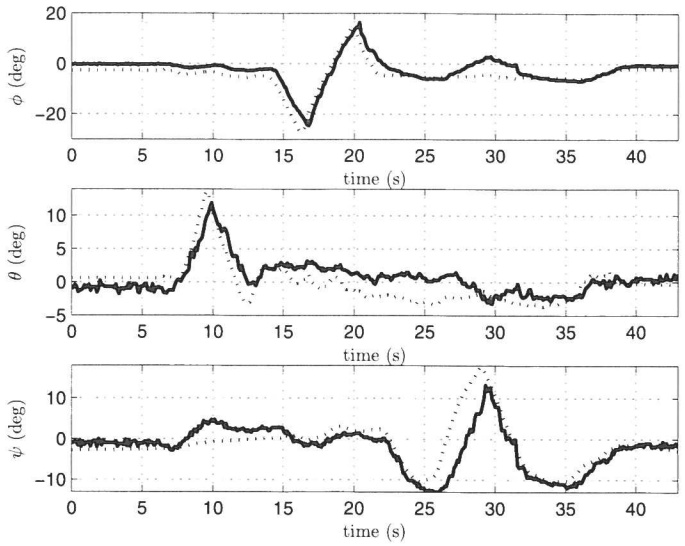


Fig. 7.9 Euler angles comparison: VO vs. IMU. *Solid lines* represent VO estimation, *dotted lines* represent IMU data

were provided by the OpenCV library, which allows performing all the tasks related to the states estimation and data fusion.

7.1.6 Experimental Results

Once the proposed sensing system was correctly installed onboard the UAV, several tests were conducted. It has been noticed that, in spite of the considerable amount of extra payload, the helicopter performance is not degraded. During such tests, the scene surrounding the helicopter was a simple unstructured indoor environment, like the one shown in Fig. 7.8.

The first experiment consisted of comparing the Euler angles estimated by VO versus the Euler angles provided by the Microbotics IMU, which are considered as the real orientation experienced by the quad-rotor. During this experiment, tilting movements of the helicopter were generated manually (by using the joystick). The Euler angles estimated by visual odometry and measured by the IMU are presented in Fig. 7.9. Next, a second experiment was conducted, consisting of a manually controlled flight of the quad-rotor over a trajectory forming a square of 150 cm × 300 cm, with a fixed altitude of 50 cm. The first movement performed consisted of 150 cm forward. Next, the quad-rotor was flown 300 cm to the right. At this point, a backwards displacement of 150 cm was performed. Finally, the quad-rotor was flown 300 cm to the left, in order to reach the same position at which it started. The system achieves an acceptable estimation of the movement described by the helicopter. The resulting estimated trajectory is presented in Fig. 7.10, this does not

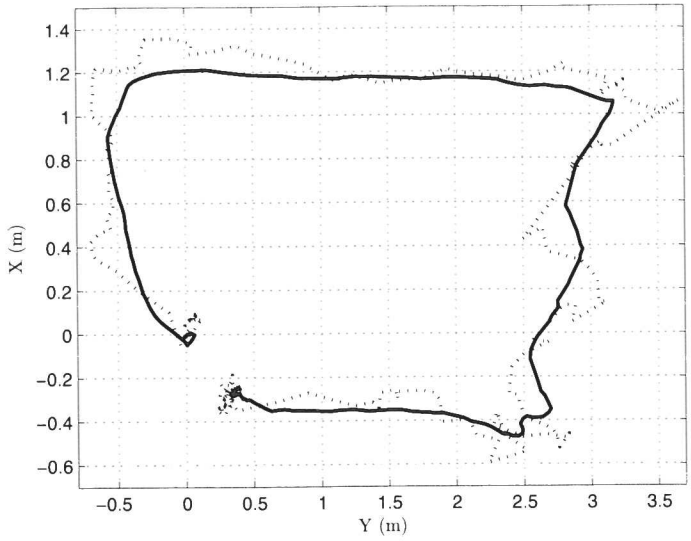


Fig. 7.10 Path that the helicopter has flown. This does not represent ground truth, it is the position estimated by the algorithm. The *dotted line* represents VO alone, *solid line* represents VO + IMU estimation

represent ground truth, it is the position estimated by the algorithm. The trajectory of 900 cm was performed in 45 seconds of flight approximatively.

Finally, a third experiment was performed, consisting of a manually controlled flight of the quad-rotor, where the helicopter altitude was varied from 50 cm to 200 cm. The idea of performing this test consists of verifying how reliable is the combination of imaging, inertial and altitude sensing system for estimating the quad-rotor altitude. Results of this experiment are shown in Fig. 7.11. The altitude estimated by means of the data fusion (continuous line) presents a similar behavior to the altitude estimated by the ultrasonic sensor (dotted line), however, still some differences exist between them. It is supposed that differences are induced by outliers in the VO algorithm, and we are still working to solve such issues.

During the previous experiments, data from the proposed sensing system were estimated at a rate of 13 Hz. From earlier work concerning quad-rotor applications (see for example [37]), it is known that such working frequency is appropriate for position control purposes. Real-time experiments consisting of autonomous hover and manually controlled flight of the quad-rotor can be seen in http://www.youtube.com/watch?v=IbTCfq_m7wc.

7.1.7 Final Comments

This section presented the development of a quad-rotor UAV equipped with a visual, inertial and altitude sensing system, which enables the helicopter to fully estimate

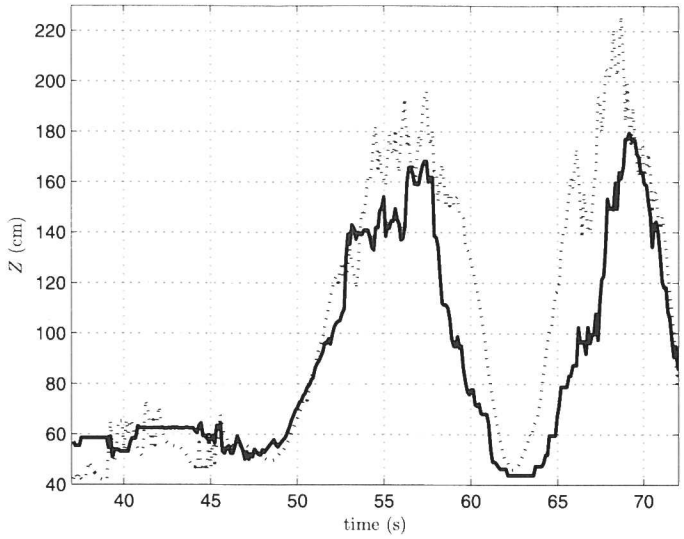


Fig. 7.11 Real-time altitude estimation. Ultrasonic sensor (*dotted line*), imaging, inertial, and altitude combination (*continuous line*)

its states without using GPS or artificial visual landmarks. In the proposed approach, a Kalman filter provided by the OpenCV library was implemented to combine VO measurements, IMU acceleration data and altitude sensor signals, with the purpose of providing accurate estimations of the states describing the behavior of the platform. The system performance was tested indoors, under real-time flight conditions.

Experimental results obtained during manually controlled flights have shown that the proposed system is capable of determining the quad-rotor 3-dimensional position and translational velocities in an accurate way. Such information is estimated at a rate of 13 Hz, which is adequate for real-time control purposes.

7.2 Comparison of Different State Estimation Algorithms for Quad-Rotor Control

The quad-rotor control requires knowledge of the state of the aircraft, mainly linear and angular velocity and position. Normally, miniature UAVs payload is severely restricted to avoid unnecessary energy consumption and increase autonomy. Therefore there is a limitation on the number and the weight of onboard sensors. As a consequence, it is crucial to develop efficient state observers for UAVs. In fact, in most of the literature on small flying robots control, the state is assumed to be completely available, see for example [27]. Standard onboard inertial sensors for UAVs are accelerometers and gyros which measure angular velocity. Small size GPS can be used to estimate the position and velocity of a UAV, however, the measurements

are reliable only when the sensor is far from urban areas. In this section, how to efficiently combine inertial measurements and imaging sensors is studied, with the purpose of effectively estimating the states of a UAV.

7.2.1 Introduction

Several research works deal with estimating the states of an aircraft. A method for obtaining the velocity of a quad-rotor UAV from acceleration measurements is presented in [12]. This approach, based on an adaptive observer technique, is compared with an extended Kalman filter in [11]. Although very promising, this technique leads to a high order observer with high computational requirements that is not suited for real-time experimental applications. Following a similar approach, a state observer of order equal to the dimension of the state vector is presented in [19]. The estimator's performance, presented in simulations, suggest that this approach can be applied in real-time applications. In [67], the authors propose a vision-based method to compute the translational speed, in the X–Y plane, of a UAV equipped with eight rotors. Similarly, a vision-based optical flow method was implemented in [37] for the estimation of the translational velocity of a quad-rotor. The two previous approaches have been validated with real-time experiments consisting of stabilized hover flights. A different approach, consisting of fusion of vision and inertial measurements is presented in [22] and [30]. This method, based on complementary filtering, provides an estimation of the vertical position and velocity of a robotic helicopter of 8 kg and a length of 1.8 m. A realistic approach for the estimation of position and velocity of a fixed-wing UAV equipped with infrared camera and inertial sensors is presented in [43]. This method is suitable to deal with instants of time when the vision system is unable to provide accurate data.

Unlike previous work which considers acceleration or vision measurements for estimating velocity, the objective here is including both visual and inertial information in the estimators. Furthermore, the implementation of such estimators is intended for a quad-rotor mini-UAV platform. Taking into account the restrictions imposed by the payload capacity and power consumption of a small UAV, the "Improved X-Flyer" quad-rotor shown in Fig. 7.12 has been equipped with an imaging and inertial sensing system consisting of a stereo rig and an IMU. These passive sensors installed onboard can provide the relative position of the quad-rotor with respect to an inertial frame, as well as the translational acceleration experienced by the vehicle. The proposed sensing system provides an appropriate framework for the development of state observers and complementary filters that combine visual and inertial information, in order to provide an accurate estimation of the position and velocity of the quad-rotor.

The goal of this research consists of identifying the most effective approach for the combination of imaging and inertial information, in order to obtain an accurate estimation of the translational velocity of the helicopter. Three of the most commonly used state estimators found in the literature have been chosen: Luenberger

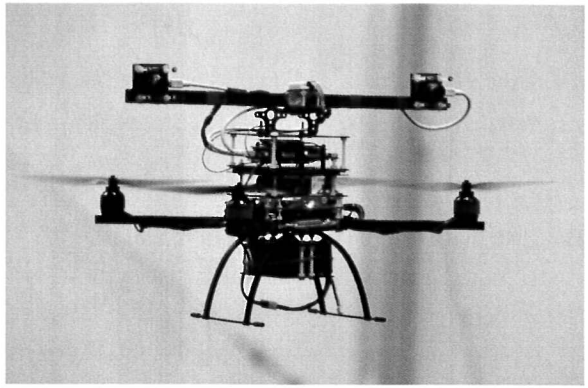


Fig. 7.12 "Improved X-Flyer" equipped with imaging and inertial sensors

observer, Kalman filter and complementary filtering. The estimators performance was evaluated in real-time experiments, consisting of hover flight and position stabilization of the quad-rotor in unstructured indoors environments. Experimental results have shown that, even though the three methodologies achieve an acceptable estimation of the vehicle's position and velocity, the helicopter controller reacts differently according to the applied estimator.

7.2.2 Problem Statement

Consider a quad-rotor hovering in an indoor unstructured environment. The quad-rotor is equipped with an IMU and analog rate gyros that directly measures the Euler angles and the angular rates, respectively. Such information can be applied in a control strategy for stabilizing the platform attitude. The vehicle is also equipped with a portable imaging and inertial sensing system. The visual part provides the 3-dimensional position of the helicopter with respect to a fixed reference frame (see Sect. 7.1.4), while the inertial part of the system provides the acceleration of the vehicle's center of mass. The objective of this study consists of using the information provided by both the imaging and inertial sensing system, in three different approaches to obtain the most accurate estimate of the helicopter translational velocity. The UAV communicates with the supervisory ground station, see Fig. 7.5, where the sensors fusion takes place.

7.2.2.1 Measurement Model

Taking into account the measured variables, and given that the stereo cameras and IMU move together as a rigid body, the discrete-time measurement model can be written as [57]

$$\mathbf{x}[k+1] = A\mathbf{x}[k] \tag{7.27}$$

$$\mathbf{y}[k] = C\mathbf{x}[k] \tag{7.28}$$

where the system states and outputs are represented by

$$\mathbf{x}[k] = [x \quad \dot{x} \quad \ddot{x} \quad y \quad \dot{y} \quad \ddot{y} \quad z \quad \dot{z} \quad \ddot{z}]^{\mathrm{T}} \tag{7.29}$$

$$\mathbf{y}[k] = [x \quad \ddot{x} \quad y \quad \ddot{y} \quad z \quad \ddot{z}]^{\mathrm{T}} \tag{7.30}$$

The state matrix of the measurement model is

$$A = \begin{bmatrix} A_x & 0_A & 0_A \\ 0_A & A_y & 0_A \\ 0_A & 0_A & A_z \end{bmatrix} \tag{7.31}$$

where

$$A_{x,y,z} = \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}; \qquad 0_A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The output matrix of the measurement model is

$$C[k] = \begin{bmatrix} C_x & 0_C & 0_C \\ 0_C & C_y & 0_C \\ 0_C & 0_C & C_z \end{bmatrix} \tag{7.32}$$

where

$$C_{x,y,z} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \qquad 0_C = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The available measurements in the above model are the position (x, y, z) and the acceleration $(\ddot{x}, \ddot{y}, \ddot{z})$. Using these measurements in a state observer, an estimation of the translational velocity can be computed. For the construction of the state observers, the model must satisfy the observability condition. This is satisfied since the system observability matrix is full rank.

7.2.3 *Design of Imaging-Inertial State Observers*

This section presents the development of three state observers for the estimation of the quad-rotor translational velocity. The estimation is obtained by fusing imaging and inertial data.

7.2.3.1 Luenberger State Observer

The proposed state observer for the model (7.27)–(7.28) is a classical discrete-time Luenberger observer [28]. The variables of the state observer are denoted by $\hat{\mathbf{x}}$ and

$\hat{\mathbf{y}}$ and are defined by

$$\hat{\mathbf{x}}[k+1] = A\hat{\mathbf{x}}[k] - L\big(\hat{\mathbf{y}}[k] - \mathbf{y}[k]\big) \tag{7.33}$$

$$\mathbf{y}[k] = C\hat{\mathbf{x}}[k] \tag{7.34}$$

The L matrix of the observer is defined as

$$L = \begin{bmatrix} L_x & 0_L & 0_L \\ 0_L & L_y & 0_L \\ 0_L & 0_L & L_z \end{bmatrix}$$

where

$$L_{x,y,z} = \begin{bmatrix} l_1 & l_2 \\ l_3 & l_4 \\ l_5 & l_6 \end{bmatrix}; \qquad 0_L = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

The observer is asymptotically stable if the error $e[k] = \hat{\mathbf{x}}[k] - \mathbf{x}[k]$ converges to zero when $k \to \infty$. The error for a Luenberger observer satisfies $e[k+1] = (A - LC)e[k]$. Choosing the eigenvalues such that the matrix $(A - LC)$ is Hurwitz, the Luenberger observer for this discrete-time system is therefore asymptotically stable. Finally, the equations describing state estimations are denoted by

$$\hat{x}[k+1] = \hat{x}[k] + T\dot{\hat{x}}[k] + \frac{T^2}{2}\ddot{\hat{x}}[k] - l_1\big(\hat{x}[k] - x[k]\big) - l_2\big(\ddot{\hat{x}}[k] - \ddot{x}[k]\big) \tag{7.35}$$

$$\dot{\hat{x}}[k+1] = \dot{\hat{x}}[k] + T\ddot{\hat{x}}[k] - l_3\big(\hat{x}[k] - x[k]\big) - l_4\big(\ddot{\hat{x}}[k] - \ddot{x}[k]\big) \tag{7.36}$$

$$\ddot{\hat{x}}[k+1] = \ddot{\hat{x}}[k] - l_5\big(\hat{x}[k] - x[k]\big) - l_6\big(\ddot{\hat{x}}[k] - \ddot{x}[k]\big) \tag{7.37}$$

$$\hat{y}[k+1] = \hat{y}[k] + T\dot{\hat{y}}[k] + \frac{T^2}{2}\ddot{\hat{y}}[k] - l_1\big(\hat{y}[k] - y[k]\big) - l_2\big(\ddot{\hat{y}}[k] - \ddot{y}[k]\big) \tag{7.38}$$

$$\dot{\hat{y}}[k+1] = \dot{\hat{y}}[k] + T\ddot{\hat{y}}[k] - l_3\big(\hat{y}[k] - y[k]\big) - l_4\big(\ddot{\hat{y}}[k] - \ddot{y}[k]\big) \tag{7.39}$$

$$\ddot{\hat{y}}[k+1] = \ddot{\hat{y}}[k] - l_5\big(\hat{y}[k] - y[k]\big) - l_6\big(\ddot{\hat{y}}[k] - \ddot{y}[k]\big) \tag{7.40}$$

$$\hat{z}[k+1] = \hat{z}[k] + T\dot{\hat{z}}[k] + \frac{T^2}{2}\ddot{\hat{z}}[k] - l_1\big(\hat{z}[k] - z[k]\big) - l_2\big(\ddot{\hat{z}}[k] - \ddot{z}[k]\big) \tag{7.41}$$

$$\dot{\hat{z}}[k+1] = \dot{\hat{z}}[k] + T\ddot{\hat{z}}[k] - l_3\big(\hat{z}[k] - z[k]\big) - l_4\big(\ddot{\hat{z}}[k] - \ddot{z}[k]\big) \tag{7.42}$$

$$\ddot{\hat{z}}[k+1] = \ddot{\hat{z}}[k] - l_5\big(\hat{z}[k] - z[k]\big) - l_6\big(\ddot{\hat{z}}[k] - \ddot{z}[k]\big) \tag{7.43}$$

where $\dot{\hat{x}}$, $\dot{\hat{y}}$ and $\dot{\hat{z}}$, represent the estimated velocities, required in the quad-rotor control strategy.

7.2.3.2 Kalman Filter

The second approach for combining vision and inertial data is the discrete Kalman filter. Let us rewrite the measurement model (7.27)–(7.28) in the form

$$\mathbf{x}[k+1] = \Phi\mathbf{x}[k] + \mathbf{w}[k] \tag{7.44}$$

$$\mathbf{z}[k] = H\mathbf{x}[k] + \mathbf{v}[k] \tag{7.45}$$

where $\mathbf{x} \in \mathbb{R}^{9\times 1}$ (equation (7.29)) is the state vector, $\Phi \in \mathbb{R}^{9\times 9}$ is the state transition matrix, $\mathbf{w} \in \mathbb{R}^{9\times 1}$ is a white sequence having known covariance structure, $\mathbf{z} \in \mathbb{R}^{6\times 1}$ is the measurement vector, $H \in \mathbb{R}^{6\times 9}$ is a matrix giving the ideal connection between the measurement and the state vector, $\mathbf{v} \in \mathbb{R}^{6\times 1}$ is the measurement error assumed to be a white sequence with known covariance structure.

The first part of the Kalman process consists of assuming one has an *a priori* estimate of the states, denoted as

$$\hat{\mathbf{x}}[k]^- = \Phi\hat{\mathbf{x}}[k-1] + \mathbf{w}[k]$$

Let us also assume knowledge of the error covariance matrix associated with $\hat{\mathbf{x}}[k]^-$, denoted as

$$P[k]^- = \Phi P[k-1]\Phi^{\mathrm{T}} + Q[k-1]$$

where Q is the covariance matrix for $\mathbf{w}$. With the assumption of the prior estimate, one seeks to use the measurements in $\mathbf{z}[k]$ to improve the prior estimate. Let us choose a linear blending of the noisy measurement and the *a priori* estimate as follows:

$$\hat{\mathbf{x}}[k] = \hat{\mathbf{x}}[k]^- + K[k]\big(\mathbf{z}[k] - H\hat{\mathbf{x}}[k]^-\big)$$

where $\mathbf{x}[k]$ is the updated estimate and $K[k]$ is the blending factor, usually known as the *Kalman gain*. A particular $K[k]$ must be identified in order to obtain an update estimate that is optimal in some sense. The solution for the optimal gain can be expressed as

$$K[k] = P[k]^- H^{\mathrm{T}}\big(HP[k]^- H^{\mathrm{T}} + R[\mathbf{k}]\big)^{-1}$$

where R is the covariance matrix for $\mathbf{v}$. The covariance matrix associated with the optimal estimate may now be computed as

$$P[k] = \big(I - K[k]H\big)P[k]^-$$

Finally, the updated estimated $\hat{\mathbf{x}}$ is projected ahead using the transition matrix

$$\hat{\mathbf{x}}[k+1]^- = \Phi\hat{\mathbf{x}}[k]$$

with its corresponding covariance matrix

$$P[k+1]^- = \Phi P[k]\Phi^{\mathrm{T}} + Q[k]$$

This process yields the required values at time $[k+1]$, and the measurement $\mathbf{z}[k+1]$ can be assimilated just as in the previous step. Further details can be found in [20] and [21].

7.2.3.3 Complementary Filter

The complementary filtering technique is the third approach used for data fusion. The definition of a complementary filter refers to the use of two or more transfer functions, which are mathematical complements of one another. For the present studies, the position estimated by means of the stereo vision system is combined with the IMU acceleration measurements in order to estimate the quad-rotor velocities $\dot{\hat{\mathbf{x}}} = (\dot{x}, \dot{y}, \dot{z})$. Let us use the next structure of the complementary filter:

$$\dot{\hat{\mathbf{x}}} = G(s)\hat{\mathbf{x}}_D + \big(1 - G(s)\big)\hat{\mathbf{x}}_I \tag{7.46}$$

where $\hat{\mathbf{x}}_D = (\dot{x}_D, \dot{y}_D, \dot{z}_D)$ are velocity estimates obtained by directly differentiating the position obtained from the imaging system, and $\hat{\mathbf{x}}_I = (\dot{x}_I, \dot{y}_I, \dot{z}_I)$ are velocities obtained by integrating the translational acceleration data. $G(s)$ is typically a low-pass filter of the form

$$G(s) = \frac{\alpha}{s+\alpha}$$

and its complement

$$1 - G(s) = \frac{s}{s+\alpha}$$

would be a high-pass filter. This is advantageous in the case where the estimate multiplying the low-pass filter is a low-bandwidth sensor, and the measurement multiplying the high-pass filter has a DC offset. Defining $G_1(s) = G(s)$ and $G_2(s) = 1 - G(s)$, a discretized expression for the filters can be expressed as

$$G_1(z) = \frac{(1 - e^{-\frac{T}{\tau}})z^{-1}}{1 - e^{-\frac{T}{\tau}}z - 1}; \qquad G_2(z) = \frac{1 - z^{-1}}{1 - e^{-\frac{T}{\tau}}z - 1} \tag{7.47}$$

where $\alpha = \frac{1}{\tau}$. $\tau = \frac{1}{w_c} = \frac{1}{2\pi f_c}$ represents the time constant of the filters, and f_c represents the filter's cutoff frequency.

In the filter implementation, the velocity obtained by integrating the acceleration is expressed as

$$\hat{\mathbf{x}}_I[k+1] = e^{-\frac{T}{\tau}}\hat{\mathbf{x}}_I[k] + \left(1 - e^{-\frac{T}{\tau}}\right)\mathbf{x}_I[k] \tag{7.48}$$

whereas the velocity obtained by differentiating the position is represented by

$$\hat{\mathbf{x}}_D[k+1] = e^{-\frac{T}{\tau}}\hat{\mathbf{x}}_D[k] + \left(\mathbf{x}[k] + \mathbf{x}[k-1]\right)T \tag{7.49}$$

Finally, using (7.48) and (7.49), the complementary filter expression is obtained,

$$\hat{\mathbf{x}}[k+1] = \hat{\mathbf{x}}_I[k+1] + \hat{\mathbf{x}}_D[k+1] \tag{7.50}$$

Further details of the complementary filter can be found in [30].

7.2.4 Experimental Results

Three real-time experiments were conducted, in order to verify the performance of the position control when using the velocity data provided by the state estimators. Such experiments are explained next.

The vehicle is flown manually until it reaches 1.5 meters above the ground. At this point, the current (x, y, z) quad-rotor 3-dimensional position, computed by means of the stereo vision algorithm is registered and such data are then used as the position reference during the rest of the experiment. The state estimators working frequency is 13 Hz, which is also the rate of the position control. The control strategy implemented during experiments is the nested saturations control presented in Sect. 6.2.3.1. The parameter values used for the altitude and the yaw controller are: $k_{pz} = 0.68$, $k_{vz} = 1.6$, $k_{p\psi} = 38$, $k_{p\psi} = 1350$. The parameters used for the nested saturations controller are: $b_4 = 0.4700$, $b_3 = 0.2349$, $b_2 = 0.1174$, and $b_1 = 0.0287$.

The first state estimator tested corresponds to the Luenberger observer. The Luenberger gains applied during the experiments are: $l_1 = 0.210$, $l_2 = 0.005$, $l_3 = 0.020$, $l_4 = 0.100$, $l_5 = 0.000$, $l_6 = 0.300$. The corresponding x and y positions as well as the z-error position described by the platform, respectively, are represented in Fig. 7.13. The velocity estimation for translational displacement is represented in Fig. 7.14. Finally, the Euler angles behavior during such experiment is represented in Fig. 7.15.

The second state estimator tested was the Kalman filter. The process and the measurement errors applied during the experiment were $\mathbf{w} = 0.150$ and $\mathbf{v} = 0.100$, respectively. The corresponding x and y positions as well as the z-error position described by the platform, respectively, are represented in Fig. 7.16. The velocity estimation for translational displacement is represented in Fig. 7.17. Finally, the Euler angles behavior during such experiment is represented in Fig. 7.18.

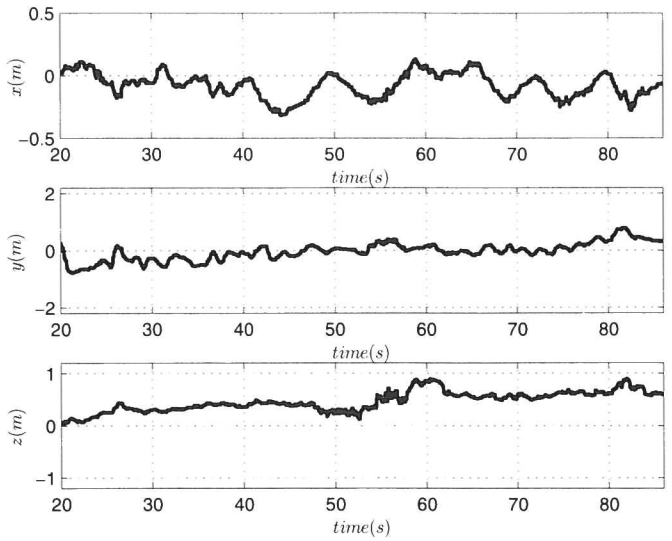


Fig. 7.13 Luenberger observer approach: x–y positions and z-error position

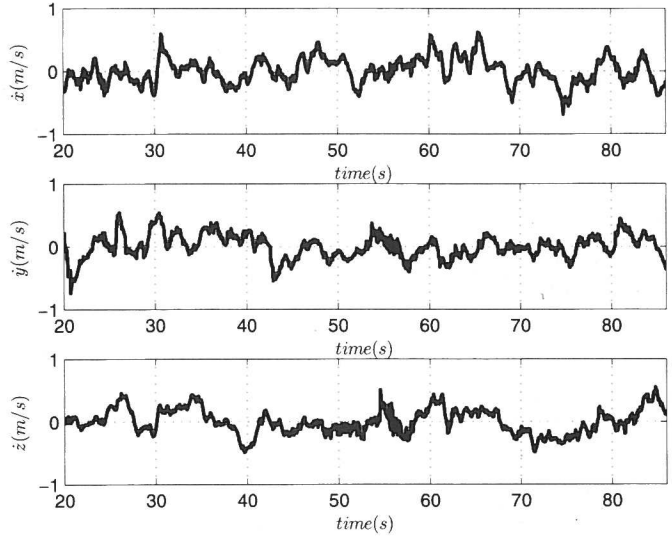


Fig. 7.14 Luenberger observer approach: x–y and z translational velocities

The third state estimator tested was the complementary filter. The corresponding x and y positions as well as the z-error position described by the platform, respectively, are shown in Fig. 7.19. The velocity estimation for translational displacement is shown in Fig. 7.20. Finally, the Euler angle behavior during such experiment is shown in Fig. 7.21.

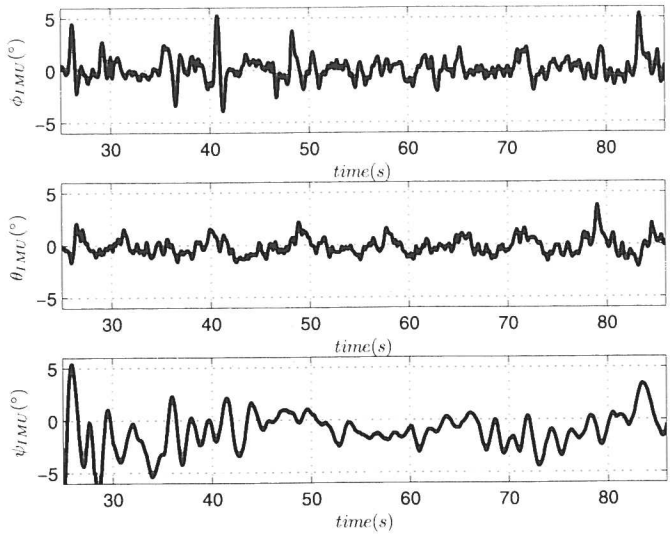


Fig. 7.15 Luenberger observer approach: Behavior of the Euler angles

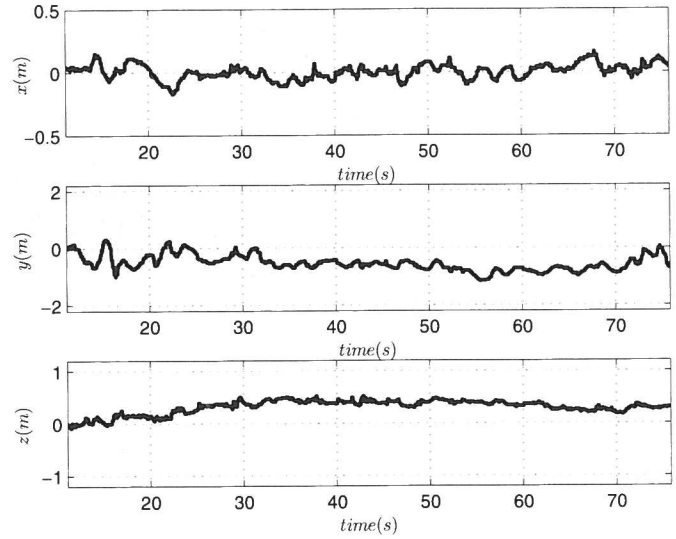


Fig. 7.16 Kalman filter approach: x–y positions and z-error position

The estimated positions of the vehicle have been analyzed, which are shown in Figs. 7.13, 7.16, and 7.19, as well as the angular behavior shown in Figs. 7.15, 7.18, and 7.21 in order to detect how the quad-rotor position control is influenced by the performance of the velocity estimators.

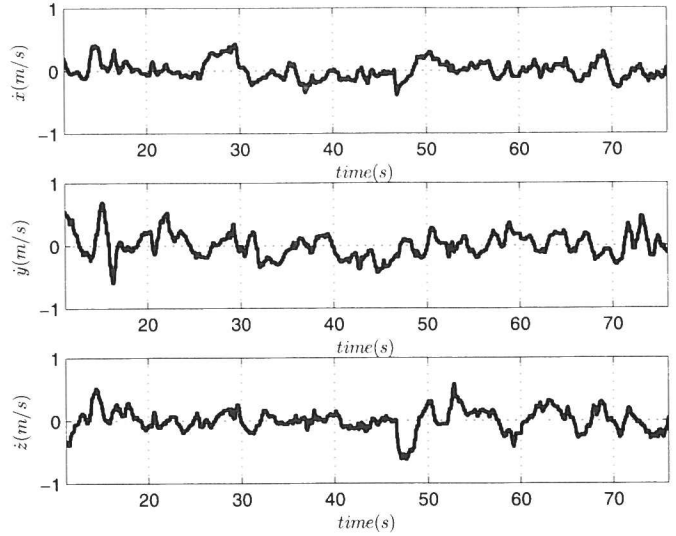


Fig. 7.17 Kalman filter approach: x–y and z translational velocities

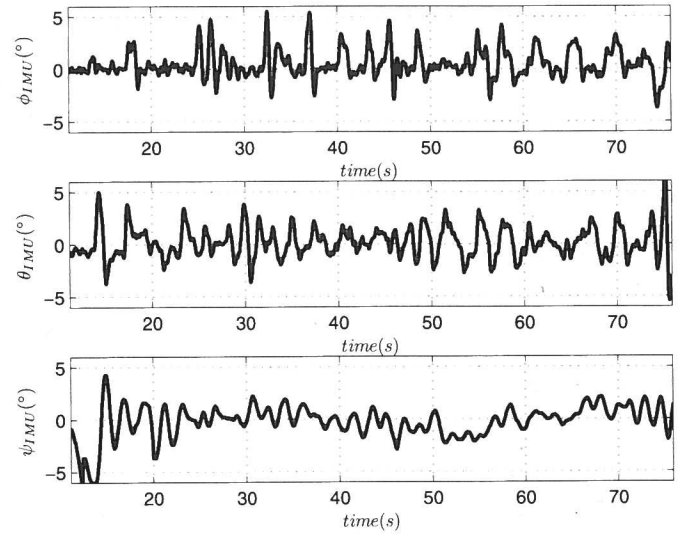


Fig. 7.18 Kalman filter approach: Behavior of the Euler angles

The velocity computed by the Luenberger observer, see Fig. 7.14, is the less-smoother signal if compared with the response of the other two estimators, it is considered that this degrades the performance of the helicopter controller. Figure 7.13 shows that the controller makes the helicopter change its x position from -0.3 m to 0.2 m. In addition, the graph shows that the quality of the altitude controller is

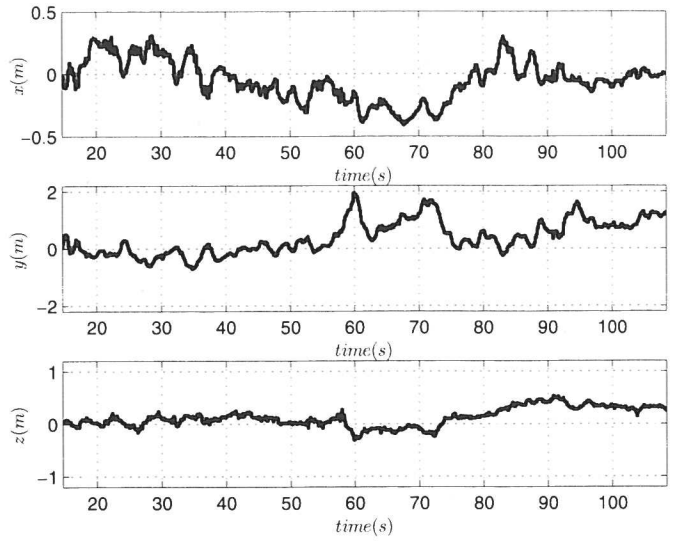


Fig. 7.19 Complementary filter approach: x–y positions and z-error position

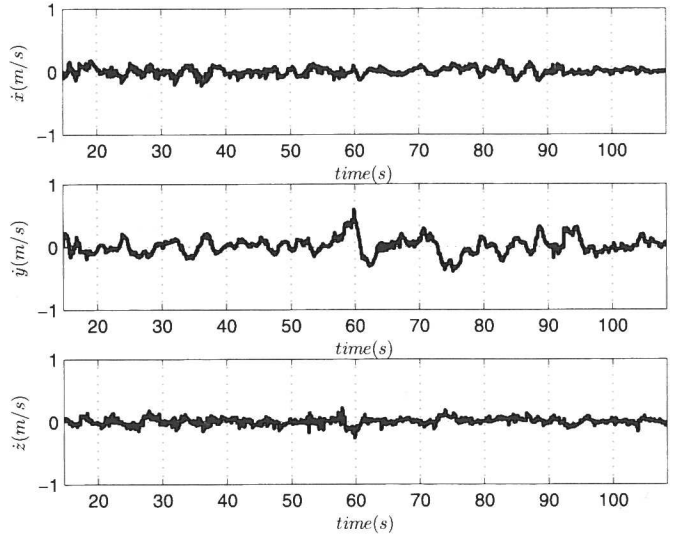


Fig. 7.20 Complementary filter approach: x–y and z translational velocities

degraded after 50 seconds, it is considered that this can be caused by energy consumption issues.

The velocity computed by the Kalman filter, see Fig. 7.17, is the smoothest velocity signal obtained from the three state estimators. Figure 7.16 shows that the controller succeeds to maintain the x position of the helicopter around the origin,

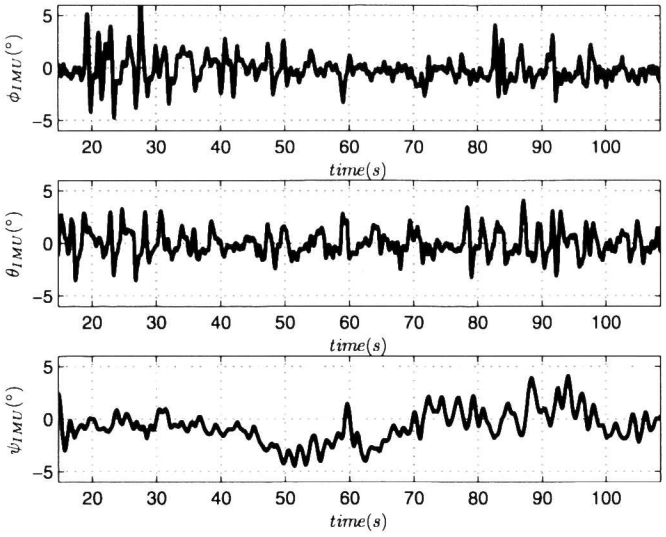


Fig. 7.21 Complementary filter approach: Behavior of the Euler angles

the y position around -0.5 m, and the altitude position with a maximum error of 0.4 m. However, the Euler angles of the platform while using this approach, see Fig. 7.18, as a result were more affected due to corrections in the position.

The controller performance when using the velocity estimated with the complementary filter (Fig. 7.20), represents a good balance between position stabilization and angular behavior. This velocity estimation is smoother than the Luenberger observer estimated velocity, however, it is evidently more attenuated than the Kalman filter estimated velocity. The position of the helicopter shown, see Fig. 7.19, was maintained, within an acceptable margin, for almost 100 seconds. This was the longest experiment achieved during our tests.

Tables 7.1 and 7.2 show the mean and standard deviation values for the position and Euler angles signals obtained during the experiments. Note that Tables 7.1 and 7.2 were computed with only one experiment for each state observer, considering the UAV in steady state. Note in these tables that the position obtained when using the Kalman filter has the lower standard deviation. Note also that when using the complementary filter the pitch and roll Euler angles are less disturbed while correcting the position. If the control objective concerns also the energy consumption, the complementary filter can be considered as a good option. An image of the quad-rotor while performing the tests can be seen in Fig. 7.22.

7.2.5 *Final Comments*

A Luenberger observer, a Kalman filter and a complementary filter were implemented and compared experimentally to estimate the translational velocity of a

Fig. 7.22 An image of the quad-rotor UAV while performing the experiments

Table 7.1 Mean values of Euler angles and position

Parameter	Luenberger observer	Kalman filter	Complementary filter
Roll angle	−0.1866°	0.4729°	0.0831°
Pitch angle	−0.0031°	0.1382°	−0.1618°
Yaw angle	−0.7076°	−0.0210°	−1.2504°
X position	−0.0774 m	0.0021 m	−0.0527 m
Y position	−0.0423 m	−0.5433 m	−0.3802 m
e_Z error	0.4634 m	−0.2157 m	0.1248 m

Table 7.2 Standard deviation of Euler angles and position

Parameter	Luenberger observer	Kalman filter	Complementary filter
Roll angle	1.3030°	1.3604°	1.1058°
Pitch angle	1.2175°	1.4999°	0.8229°
Yaw angle	1.5601°	1.1205°	1.7270°
X position	0.0992 m	0.0575 m	0.1628 m
Y position	0.3016 m	0.2905 m	0.5746 m
e_Z error	0.1878 m	0.1248 m	0.1733 m

quad-rotor UAV equipped with an imaging and inertial sensing system. The stereo vision system was used to estimate the (x, y, z) 3-dimensional position of the aerial vehicle with respect to a fixed inertial frame, while an IMU was used to provide the linear accelerations experienced by the platform. A control strategy was implemented onboard to stabilize the position of the helicopter while using the estimated position and velocity data.

Experimental results have shown that the response of the system is smoother and that the vehicle position is closer to the desired values when using the velocity estimated by the Kalman filter.

7.3 Concluding Remarks

This chapter presented the development of a quad-rotor robotic platform equipped with a visual, inertial and altitude sensing system, which allows the helicopter to fully estimate its states without using GPS or artificial visual landmarks. The research objective consists of enabling the UAV to autonomously perform take-off, relative positioning, navigation and landing, when evolving in unstructured, indoors, and GPS-denied environments.

A stereo visual odometry algorithm was implemented with the purpose of estimating the vehicle ego-motion, in all six degrees of freedom, from the images provided by the cameras. The acceleration experienced by the vehicle was measured by an IMU, and, in addition, an ultrasonic sensor provided an additional estimation of the helicopter altitude. The output of these three sensors was combined in a simple Kalman filter strategy, allowing the estimation of the quad-rotor 3-dimensional position and translational velocity in an accurate way. The effectiveness of the proposed approach was evaluated in real-time experiments indoors, achieving an acceptable estimation of the quad-rotor's translational dynamics. The vehicle's states are provided at a rate of 13 Hz, which is an adequate frequency for performing vision-based position control tasks.

Once the desired functionality of the proposed system was achieved, a Luenberger observer, a Kalman filter and a complementary filter were implemented and compared in real-time experiments, with the purpose of identifying the most effective approach for combining visual odometry with inertial measurements. By using the estimated position and velocity data, a nested saturations control strategy was implemented onboard for stabilizing the vehicle when flying. It was experimentally found that the response of the system is smoother and that the vehicle's position is closer to the desired values when using the velocity estimated by the Kalman filter.