9

Visual Servoing for UAVs

Pascual Campoy, Iván F. Mondragón,
Miguel A. Olivares-Méndez and Carol Martínez
Universidad Politécnica de Madrid (Computer Vision Group)
Spain

1. Introduction

Vision is in fact the richest source of information for ourself and also for outdoors Robotics, and can be considered the most complex and challenging problem in signal processing for pattern recognition. The first results using Vision in the control loop have been obtained in indoors and structured environments, in which a line or known patterns are detected and followed by a robot (Feddema & Mitchell (1989), Masutani et al. (1994)). Successful works have demonstrated that visual information can be used in tasks such as servoing and guiding, in robot manipulators and mobile robots (Conticelli et al. (1999), Mariottini et al. (2007), Kragic & Christensen (2002).)

Visual Servoing is an open issue with a long way for researching and for obtaining increasingly better and more relevant results in Robotics. It combines image processing and control techniques, in such a way that the visual information is used within the control loop. The bottleneck of Visual Servoing can be considered the fact of obtaining robust and on-line visual interpretation of the environment, which can be usefully treated by control structures and algorithms. The solutions provided in Visual Servoing are typically divided into Image Based Control Techniques and Pose Based Control Techniques, depending on the kind of information provided by the vision system that determine the kind of references that have to be sent to the control structure (Hutchinson et al. (1996), Chaumette & Hutchinson (2006) and Siciliano & Khatib (2008)). Another classical division of the Visual Servoing algorithms considers the physical disposition of the visual system, yielding to eye-in-hand systems and eye-to-hand systems, that in the case of Unmanned Aerial Vehicles (UAV) can be translated as on-board visual systems (Mejias (2006)) and ground visual systems (Martínez et al. (2009)).

The challenge of Visual Servoing is to be useful in outdoors and non-structured environments. For this purpose the image processing algorithms have to provide visual information that has to be robust and works in real time. UAV can therefore be considered as a challenging testbed for visual servoing, that combines the difficulties of abrupt changes in the image sequence (i.e. vibrations), outdoors operation (non-structured environments) and 3D information changes (Mejias et al. (2006)). In this chapter we give special relevance to the fact of obtaining robust visual information for the visual servoing task. In section (2).we overview the main algorithms used for visual tracking and we discuss their robustness when they are applied to image sequences taken from the UAV. In sections (3). and (4). we analyze how vision systems can perform 3D pose estimation that can be used for

Source: Visual Servoing, Book edited by: Rong-Fong Fung,
ISBN 978-953-307-095-7, pp. 234, April 2010, INTECH, Croatia, downloaded from SCIYO.COM

controlling whether the camera platform or the UAV itself. In this context, section (3). analyzes visual pose estimation using multi-camera ground systems, while section (4). analyzes visual pose estimation obtained from onboard cameras. On the other hand, section (5)., shows two position based control applications for UAVs. Finally section (6). explodes the advantages of fuzzy control techniques for visual servoing in UAVs.

2. Image processing for visual servoing

Image processing is used to find characteristics in the image that can be used to recognize an object or points of interest. This relevant information extracted from the image (called features) ranges from simple structures, such as points or edges, to more complex structures, such as objects. Such features will be used as reference for any visual servoing task and control system.

On image regions, the spatial intensity also can be considered as a useful characteristic for patch tracking. In this context, the region intensities are considered as a unique feature that can be compared using correlation metrics on image intensity patterns.

Most of the features used as reference are interest points, which are points in an image that have a well-defined position, can be robustly detected, and are usually found in any kind of images. Some of these points are corners formed by the intersection of two edges, and others are points in the image that have rich information based on the intensity of the pixels. A detector used for this purpose is the Harris corner detector (Harris & Stephens (1988)). It extracts corners very quickly based on the magnitude of the eigenvalues of the autocorrelation matrix. Where the local autocorrelation function measures the local changes of a point with patches shifted by a small amount in different directions. However, taking into account that the features are going to be tracked along the image sequence, it is not enough to use only this measure to guarantee the robustness of the corner. This means that good features to track (Shi & Tomasi (1994)) have to be selected in order to ensure the stability of the tracking process. The robustness of a corner extracted with the Harris detector can be measured by changing the size of the detection window, which is increased to test the stability of the position of the extracted corners. A measure of this variation is then calculated based on a maximum difference criteria. Besides, the magnitude of the eigenvalues is used to only keep features with eigenvalues higher than a minimum value. Combination of such criteria leads to the selection of the good features to track. Figure 1(a) shows and example of good features to track on a image obtained on a UAV.

The use of other kind of features, such as edges, is another technique that can be applied on semi-structured environments. Since human constructions and objects are based on basic geometrical figures, the Hough transform (Duda & Hart (1972)) becomes a powerful technique to find them in the image. The simplest case of the algorithm is to find straight lines in an image that can be described with the equation $y = mx + b$. The main idea of the Hough transform is to consider the characteristics of the straight line not as image points x or y, but in terms of its parameters m and b, representing the same line as $y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right)$ in the parameter space, that is based on the angle of the vector from the origin to this closest point on the line (θ) and distance between the line and the origin (r). If a set of points form a straight line, they will produce sinusoids that cross at the parameters of that line. Thus, the problem of detecting collinear points can be converted to the problem of finding concurrent curves. To apply this concept just to points that might be on a line, some pre-processing algorithms are used to find edge features, such as the Canny

edge detector (Canny (1986)) or the ones based on derivatives of the images obtained by a convolution of image intensities and a mask (Sobel I. (1968)). These methods have been used in order to find power lines and isolators in an UAV inspection application (Mejías et al. (2007)).
The problem of tracking features can be solved with different approaches. The most popular algorithm to track features and image regions, is the Lucas-Kanade algorithm (Lucas & Kanade (1981)) which have demonstrated a good performance for real time with a good stability for small changes. Recently, feature descriptors have been successfully applied on visual tracking, showing a good robustness for image scaling, rotations, translations and illumination changes, eventhough they are time expensive to calculate. The generalized Lucas Kanade algorithm is overviewed on subsection 2.1, where it is applied for patch tracking and also for optical flow calculation, using the sparse L-K (subsection 2.1.1) and pyramidal L-K (subsection 2.1.2) variations. On subsection 2.2, features descriptors are introduced and used for robust matching, as explained on subsection 2.3

2.1 Appearance tracking

Appearance-based tracking techniques does not use features. They use the intensity values of a 'patch' of pixels that correspond to the object to be tracked. The method to track this patch of pixels is the generalized L-K algorithm, that works under three premises: first, the intensity constancy: the vicinity of each pixel considered as a feature does not change as it is tracked from frame to frame; second, the change in the position of the features between two consecutive frames must be minimum, so that the features are close enough to each other; and third, the neighboring points move in a solidarity form and have spatial coherence.
The patch is related to the next frame by a warping function that can be the optical flow or another model of motion. Taking into account the previously mentioned L-K premisses, the problem can be formulated in this way: lets define X as the set of points that form the patch window or template image T, where $\mathbf{x} = (x,y)^T$ is a column vector with the coordinates in the image plane of a given pixel and $T(\mathbf{x}) = T(x,y)$ is the grayscale value of the images a the locations $\mathbf{x}$. The goal of the algorithm is to align the template T with the input image I (where $I(\mathbf{x}) = I(x,y)$ is the grayscale value of the images a the locations $\mathbf{x}$). Because T transformed must match with a sub-image of I, the algorithm will find the set of parameters $\mu = (\mu_1,\mu_2, ...\mu_n)$ for a motion model function (e.g., Optical Flow, Affine, Homography) $W(\mathbf{x};\mu)$, also called the warping function. The objective function of the algorithm to be minimized in order to align the template and the actual image is equation 1:

$$e(\mathbf{W}) = \sum_{\forall \mathbf{x} \epsilon X} (I(W(\mathbf{x};\mu) - T(\mathbf{x}))^2 w(\mathbf{x}) \tag{1}$$

where $w(\mathbf{x})$ is a function to assign different weights to the comparison window. In general $w(\mathbf{x}) = 1$. Alternatively, w could be a Gaussian function to emphasize the central area of the window. This equation can also be reformulated to make it possible to solve for track sparse feature as is explained on section 2.1.1.
The Lucas Kanade problem is formulated to be solved in relation to all features in the form of a least squares' problem, having a closed form solution as follows.
Defining $w(\mathbf{x}) = 1$, the objective function (equation 1) is minimized with respect to μ and the sum is performed over all of the pixels $\mathbf{x}$ on the template image. Since the minimization process has to be made with respect to μ, and there is no lineal relation between the pixel

position and its intensity value, the Lucas-Kanade algorithm assumes a known initial value for the parameters μ and finds increments of the parameters $\delta\mu$. Hence, the expression to be minimized is:

$$\sum_{\forall \mathbf{x} \epsilon X} (I(W(\mathbf{x};\mu + \delta\mu) - T(\mathbf{x}))^2 \tag{2}$$

and the parameter actualization in every iteration is $\mu = \mu + \delta\mu$. In order to solve equation 2 efficiently, the objective function is linearized using a Taylor Series expansion employing only the first order terms. The parameter to be minimized is $\delta\mu$. Afterwards, the function to be minimized looks like equation 3 and can be solved like a "least squares problem" with equation 4.

$$\sum_{\forall \mathbf{x} \epsilon X} (I(W(\mathbf{x};\mu) + \nabla I \frac{\partial W}{\partial \mu} \delta\mu - T(\mathbf{x}))^2 \tag{3}$$

$$\delta\mu = H^{-1} \sum_{\forall \mathbf{x} \epsilon X} (\nabla I \frac{\partial W}{\partial \mu})^T (T(\mathbf{x}) - I(W(\mathbf{x};\mu)) \tag{4}$$

where H is the Hessian Matrix approximation,

$$H = \sum_{\forall \mathbf{x} \epsilon X} (\nabla I \frac{\partial W}{\partial \mu})^T (\nabla I \frac{\partial W}{\partial \mu}) \tag{5}$$

More details about this formulation can be found in (Buenaposada et al. (2003) and Baker and Matthews (2002)), where some modifications are introduced in order to make the minimization process more efficient, by inverting the roles of the template and changing the parameter update rule from an additive form to a compositional function. This is the so called ICIA (Inverse Compositional Image Alignment) algorithm, first proposed in (Baker and Matthews (2002)). These modifications where introduced to avoid the cost of computing the gradient of the images, the Jacobian of the Warping function in every step and the inversion of the Hessian Matrix that assumes the most computational cost of the algorithm.

2.1.1 Sparse Lucas Kanade

The Lucas Kanade algorithm can be applied on small windows around distinctive points as a sparse technique. In this case, the template is a small window (i.e., size of 3, 5, 7 or 9 pixels) and the warping function is defined by only a pure translational vector. In this context, the first assumption of the Lucas-Kanade method can be expressed as given a point $\mathbf{x}_i = (x, y)$ at time t which intensity is $I(x, y, t)$ will have moved by v_x, v_y and Δt between the two image frames, the following equation can be formulated:

$$I(x,y,t) = I(x + v_x, y + v_x, \Delta t) \tag{6}$$

If the general movement can be consider small and using the Taylor series, equation 6 can be developed as:

$$I(x + v_x, y + v_y, t + \Delta t) = I(x,y,t) + \frac{\partial I}{\partial x} v_x + \frac{\partial I}{\partial y} v_y + \frac{\partial I}{\partial t} \Delta t + H.O.T. \tag{7}$$

Because the higher order terms *H.O.T.* can being ignored, from equation we found that:

$$\frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y + \frac{\partial I}{\partial t}\Delta t = 0 \tag{8}$$

where v_x,v_y are the x and y components of the velocity or optical flow of $I(x,y,t)$ and $I_x = \frac{\partial I}{\partial x}$, $I_y = \frac{\partial I}{\partial y}$ and $I_t = \frac{\partial I}{\partial t}$ are the derivatives of the image at point $p = (x,y,t)$

$$I_x v_x + I_y v_y = -I_t \tag{9}$$

Equation 9 is known as the *Aperture Problem* of the optical flow. It arises when you have a small aperture or window in which to measure motion. If motion is detected in this small aperture, it is often that it will be seeing as a edge and not as a corner, causing that the movement direction can not be determined. To find the optical flow another set of equations is needed, given by some additional constraint.

The Lucas-Kanade algorithm forms the additional set of equation assuming that there is a local small window of size $m \times m$ centered at point $p = (x,y)$ in which all pixels moves coherently. If the windows pixel are numerates as 1...n, with $n = m^2$, a set of equations can be found:

$$\begin{aligned} I_{x_1}v_x + I_{y_1}v_y &= -I_{t_1} \\ I_{x_2}v_x + I_{y_2}v_y &= -I_{t_2} \\ &\vdots \\ I_{x_n}v_x + I_{y_n}v_y &= -I_{t_n} \end{aligned} \tag{10}$$

Equation 10 have more than two equations for the two unknowns and thus the system is over-determined. A systems of the form $\mathbf{Ax} = \mathbf{b}$ can be former as equation 12 shows.

$$\mathbf{A} = \begin{bmatrix} I_{x_1} & I_{y_1} \\ I_{x_2} & I_{y_2} \\ \vdots & \vdots \\ I_{x_n} & I_{y_n} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} v_x \\ v_y \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -I_{t_1} \\ -I_{t_2} \\ \vdots \\ -I_{t_n} \end{bmatrix}$$

or

$$\begin{bmatrix} I_{x_1} & I_{y_1} \\ I_{x_2} & I_{y_2} \\ \vdots & \vdots \\ I_{x_n} & I_{y_n} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} -I_{t_1} \\ -I_{t_2} \\ \vdots \\ -I_{t_n} \end{bmatrix} \tag{11}$$

The least squares method can be used to solve the over determined system of equation 12, finding that the optical flow can be defined as:

$$\begin{aligned} \mathbf{A}^T\mathbf{A}\mathbf{x} &= \mathbf{A}^T\mathbf{b} \\ &\text{or} \\ \begin{bmatrix} v_x \\ v_y \end{bmatrix} &= (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \end{aligned} \tag{12}$$

2.1.2 Pyramidal L-K

On images with high motion, good matched features can be obtained using the Pyramidal Lucas-Kanade algorithm modification (Bouguet Jean Yves (1999)). It is used to solve the problem that arise when large and non-coherent motion are presented between consecutive frames, by firsts tracking features over large spatial scales on the pyramid image, obtaining an initial motion estimation, and then refine it by down sampling the levels of the images in the pyramid until it arrives to the original scale.

The overall pyramidal tracking algorithm proceeds as follows: first, a pyramidal representation of an image I of size $widthpixels \times heightpixels$ is generated. The zeroth level is composed by the original image and defined as I^0, then pyramids levels are recursively computed by dawnsampling the last available level (compute I^1 form I^0, then I^2 from I^1 and so on until I^{Lm} form I^{L-1})). Typical maximum pyramids Levels L_m are 2, 3 and 4. Then, the optical flow is computed at the deepest pyramid level L_m. Then, the result of that computation is propagated to the upper level $L_m - 1$ in a form of an initial guess for the pixel displacement (at level $L_m - 1$). Given that initial guess, the refined optical flow is computed at level $L_m - 1$, and the result is propagated to level $L_m - 2$ and so on up to the level 0 (the original image).

2.2 Feature descriptors and tracking

Feature description is a process to obtain interest points in the image which are defined by a series of characteristics that make it suitable for being matched on image sequences. This characteristics can include a clear mathematical definition, a well-defined position in image space and a local image structure around the interest point. This structure has to be rich in terms of local information contents that has to be robust under local and global perturbations in the image domain. These robustness includes those deformations arising from perspective transformations (i.e, scale changes, rotations and translations) as well as illumination/brightness variations, such that the interest points can be reliably computed with high degree of reproducibility.

There are many feature descriptors suitable for visual matching and tracking, from which Scale Invariant Feature Transform (SIFT) and Speeded Up Robust Feature algorithm (SURF) have been the more widely use on the literature and are overview in sections 2.2.1 and 2.2.2.

2.2.1 SIFT features

The SIFT (Scale Invariant Feature Transform) detector (Lowe (2004)) is one of the most widely used algorithms for interest point detection (called keypoints in the SIFT framework) and matching. This detector was developed with the intention to be used for object recognition. Because of this, it extracts keypoints invariant to scale and rotation using the gaussian difference of the images in different scales to ensure invariance to scale. To achieve invariance to rotation, one or more orientations based on local image gradient directions are assigned to each keypoint. The result of all this process is a descriptor associated to the keypoint, which provides an efficient tool to represent an interest point, allowing an easy matching against a database of keypoints. The calculation of these features has a considerable computational cost, which can be assumed because of the robustness of the keypoint and the accuracy obtained when matching these features. However, the use of these features depends on the nature of the task: whether it needs to be done fast or accurate. Figure 1(b) shows and example of SIFT keypoints on an aerial image taken with an UAV.

SIFT features can be used to track objects, using the rich information given by the keypoints descriptors. The object is matched along the image sequence comparing the model template (the image from which the database of features is created) and the SIFT descriptor of the current image, using the nearest neighbor method. Given the high dimensionality of the keypoint descriptor (128), its matching performance is improved using the Kd-tree search algorithm with the Best Bin First search modification proposed by Lowe (Beis and Lowe (1997)). The advantage of this method lies in the robustness of the matching using the descriptor, and in the fact that this match does not depend on the relative position of the template and the current image. Once the matching is performed, a perspective transformation is calculated using the matched Keypoints, comparing the original template with the current image.

2.2.2 SURF features

Speeded Up Robust Feature algorithm (Herbert Bay et al. (2006)) extracts features from an image which can be tracked over multiple views. The algorithm also generates a descriptor for each feature that can be used to identify it. SURF features descriptor are scale and rotation invariant. Scale invariance is attained using different amplitude gaussian filters, in such a way that its application results in an image pyramid. The level of the stack from which the feature is extracted assigns the feature to a scale. This relation provides scale invariance. The next step is to assign a repeatable orientation to the feature. The angle is calculated through the horizontal and vertical Haar wavelet responses in a circular domain around the feature. The angle calculated in this way provides a repeatable orientation to the feature. As with the scale invariance the angle invariance is attained using this relationship. Figure 1(c) shows and example of SURF features on an aerial image.

SURF descriptor is a 64 element vector. This vector is calculated in a domain oriented with the assigned angle and sized according to the scale of the feature. Descriptor is estimated using horizontal and vertical response histograms calculated in a 4 by 4 grid. There are two variants to this descriptor: the first provides a 32 element vector and the other one a 128 element vector. The algorithm uses integral images to implement the filters. This technique makes the algorithm very efficient.

The procedure to match SURF features is based on the descriptor associated to the extracted interest point. An interest point in the current image is compared to an interest point in the previous one by calculating the Euclidean distance between their descriptor vectors.

(a) (b) (c)

Fig. 1. Comparison between features point extractors. Figure 1(a) are features obtained using Good Features to Track, figure 1(b) are keypoints obtained using SIFT (the green arrows represents the keypoints orientation and scale) and figure 1(c) are descriptors obtained using SURF (red circles and line represents the descriptor scale and angle).

2.3 Robust matching

A set of corresponding or matched points between two images are frequently used to calculate geometrical transformation models like affine transformations, homographies or the fundamental matrix in stereo systems. The matched points can be obtained by a variety of methods and the set of matched points obtained often has two error sources. The first one is the measurement of the point position, which follows a Gaussian distribution. The second one is the *outliers* to the Gaussian error distribution, which are the mismatched points given by the selected algorithm. These outliers can severely disturb the estimated function, and consequently alter any measurement or application based on this geometric transformation. The goal then, is to determine a way to select a set of *inliers* from the total set of correspondences, so that the desired projection model can be estimated with some standard methods, but employing only the set of pairs considered as inliers. This kind of calculation is considered as *robust estimation*, because the estimation is tolerant (robust) to measurements following a different or unmodeled error distribution (outliers).

Thus, the objective is to filter the total set of matched points in order to detect and eliminated erroneous matched and estimate the projection model employing only the correspondences considered as inliers. There are many algorithms that have demonstrated good performance in model fitting, some of them are the Median of Squares (LMeds) (Rousseeuw & Leroy (1987)) and Random Sample Consensus (RANSAC) algorithm (Fischer & Bolles (1981)). Both are randomized algorithms and are able to cope with a large proportion of outliers.

In order to use a robust estimation method for a projective transformation, we will assume that a set of matched points between two projective planes (two images) obtained using some of the methods describe in section (2). are available. This set includes some unknown proportion of outliers or bad correspondences, giving a series of matched points $(x_i,y_i) \leftrightarrow (x'_i,y'_i)$ for $i = 1. \; . \; .n$, from which a perspective transformation must be calculated, once the outliers have been discarded.

For discard the outliers from the set of matched points, we use the RANSAC algorithm (Fischer & Bolles (1981)). It achieves its goal by iteratively selecting a random subset of the original data points by testing it to obtain the model and evaluating the model consensus, which is the total number of original data points that best fit the model. The model is obtained using a close form solution according to the desired projective transformation (an example is show on section 2.3.1). This procedure is then repeated a fixed number of times, each time producing either a model which is rejected because too few points are classified as inliers, or a refined model. When total trials are reached, the algorithm return the projection model with the largest number of inliers. The algorithm 1 shows a the general steps to obtain a robust transformation. Further description can be found on (Hartley & Zisserman (2004), Fischer & Bolles (1981)).

2.3.1 Robust homography

As an example of the generic robust method described above, we will show its application for a robust homography estimation. It can be viewed as the problem of estimating a 2D projective transformation that given a set of points $\bar{\mathbf{x}}_i$ in $\mathbb{P}^2$ and a corresponding set of points $\bar{\mathbf{x}}'_i$ in $\mathbb{P}^2$, compute the 3x3 matrix $\mathbf{H}$ that takes each $\bar{\mathbf{x}}_i$ to $\bar{\mathbf{x}}'_i$ or $\bar{\mathbf{x}}'_i = \mathbf{H}\bar{\mathbf{x}}_i$. In general the points $\bar{\mathbf{x}}_i$ and $\bar{\mathbf{x}}'_i$ are points in two images or in 2D plane surfaces.

Algorithm 1 Projective Transformation estimation using RANSAC

Require: Set of matched points $\mathbf{x}_i = (x_i, y_i) \leftrightarrow \mathbf{x'}_i = (x'_i, y'_i)$ for $i = 1 \ldots n$
Define $s =$ Minimum set of points to estimate the minimal solution.
Define $p =$ Probability that al least one of the random samples is free form outliers
Define $t =$ distance threshold to consider a point as an inlier for some model.
Define $\epsilon =$ Initial probability that any selected point is an outlier.
Define $Concesus =$ Desired number of minimum Inliers based on the total number of matched points
Calculate the maximum number os samples $N = \log(1-p)/\log(1-(1-\epsilon)^s)$
while $N > Trials$ **do**
  Randomly select s pairs of matched points
  Calculate the minimal solution for the model under test, using selected s points
  $inliers = 0$
  for $i = 0$ to n **do**
    Calculate the distance $d^2_{transfer} = d(\mathbf{x'}_i, \mathbf{H}\mathbf{x}_i)^2 + d(\mathbf{x}_i, \mathbf{H}^{-1}\mathbf{x'}_i)^2$
    if $d_{transfer} < t$ **then**
      $inliers = inliers + 1$
    end if
  end for
  if $inliers > Concensus$ **then**
    Calculate the final projective transformation using all inliers points
    $Concensus = inliers$
  end if
  recalculate $\epsilon = 1 - (inliers/n)$
  recalculate $N = \log(1-p)/\log(1-(1-\epsilon)^s)$
  $Trials = Trials + 1$
end while

Taking into account that the number of degrees of freedom of the projective transformation is eight (defined up to scale) and because each point to point correspondences $(x_i, y_i) \leftrightarrow (x'_i, y'_i)$ gives rise to two independent equations in the entries of $\mathbf{H}$, is enough with four correspondences to have a exact solution or minimal solution. If more than four points correspondences are given, the system is over determined and $\mathbf{H}$ is estimated using a minimization method. So, in order to use the algorithm 1, we define the minimum set of points to be $s = 4$.

If matrix $\mathbf{H}$ is written in the form of a vector $\mathbf{h} = [h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33}]^t$ the homogeneous equations $\bar{\mathbf{x}}' = \mathbf{H}\bar{\mathbf{x}}$ for n points could be formed as $\mathbf{Ah} = 0$, with $\mathbf{A}$ a $2n \times 9$ matrix defined by equation 13:

$$A = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 & -y'_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_nx'_n & -y_nx'_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_ny'_n & -y_ny'_n & -y'_n \end{bmatrix} \tag{13}$$

In general, equation 13 can be solved using three different methods (the inhomogeneous solution, the homogeneous solution and non-linear geometric solution) as explained in

Criminisi et al. (1999). The most widely use of these methods is the inhomogeneous solution. In this method, one of the nine matrix elements is given a fixed unity value, forming an equation of the form $\mathbf{A'h'} = \mathbf{b}$ as is shown in equation 14.

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_nx_n' & -y_nx_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_ny_n' & -y_ny_n' \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x_1' \\ y_1' \\ \vdots \\ x_n' \\ y_n' \end{bmatrix} \tag{14}$$

The resulting simultaneous equations for the 8 unknown elements are then solved using a Gaussian elimination in the case of a minimal solution or using a pseudo-inverse method in case of an over-determined system Hartley and Zisserman (2004).
Figure 2 shows an example of a car tracking using a UAV, in which SURF algorithm, is used to obtain visual features, and the RANSAC algorithm is used for outliers rejection.

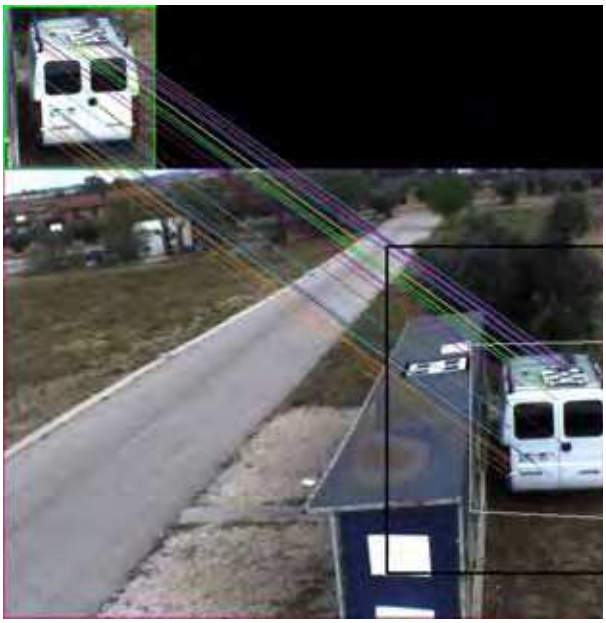

Fig. 2. Robust Homography Estimation using SURF features on a car tracking from a UAV. Up: Reference template. Down: Scene view, in which are present translation, rotation, and occlusions.

3. Ground visual system for pose estimation

Multi-camera systems are considered attractive because of the huge amount of information that can be recovered and the increase of the camera FOV (Field Of View) that can be

obtained with these systems. These characteristics can help solving common vision problems such as occlusions, and can offer more tools for control, tracking, representation of objects, object analysis, panoramic photography, surveillance, navigation of mobile vehicles, among other tasks. However, in spite of the advantages offered by these systems, there are some applications where the hardware and the computational requirements make a multi-camera solution inadequate, taking into account that the larger the number of cameras used, the greater the complexity of the system is.

For example, in the case of pose estimation algorithms, when there is more than one camera involved, there are different subsystems that must be added to the algorithm:

- Camera calibration
- Feature Extraction and tracking in multiple images
- Feature Matching
- 3D reconstruction (triangulation)

Nonetheless, obtaining an adequate solution for each subsystem, it could be possible to obtain a multiple view-based 3D position estimation at real-time frame rates.

This section presents the use of a multi-camera system to detect, track, and estimate the position and orientation of a UAV by extracting some onboard landmarks, using the triangulation principle to recovered their 3D location, and then using this 3D information to estimate the position and orientation of the UAV with respect to a *World Coordinate System*. This information will be use later into a UAV's control loop to develop positioning and landing tasks.

3.0.2 Coordinate systems

Different coordinate systems are used to map the extracted visual information from $\Re^2$ to $\Re^3$, and then to convert this information into commands to the helicopter. This section provides a description of the coordinate systems and their corresponding transformations to achieve vision-based tasks.

There are different coordinate systems involved: the *Image Coordinate System* (X_i), that includes the *Lateral* (X_f) and *Central Coordinate Systems* (X_u) in the image plane, the *Camera Coordinate System* (X_c), the *Helicopter Coordinate System* (X_h), and an additional one: the *World Coordinate System* (X_w), used as the principal reference system to control the vehicle (see figure 3).

- ***Image* and *Camera Coordinate Systems***

The relation between the *Camera Coordinate System* and the *Image Coordinate System* is taken from the "*pinhole*" camera model. It states that any point referenced in the *Camera Coordinate System* $\mathbf{x_c}$ is projected onto the image plane in the point $\mathbf{x_f}$ by intersecting the ray that links the 3D point $\mathbf{x_c}$ with the center of projection and the image plane. This mapping is described in equation15, where $\mathbf{x_c}$ and $\mathbf{x_f}$ are represented in homogenous coordinates.

$$\begin{bmatrix} nx_f \\ ny_f \\ n \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \qquad (15)$$

$$\mathbf{x_f} = \mathbf{K^k}[\mathbf{I}|\mathbf{0}]\mathbf{x_c}$$

The matrix $\mathbf{K^k}$ contains the intrinsic camera parameters of the k^{th} camera, such as the coordinates of the center of projection (c_x, c_y) in pixel units, and the focal length (f_x, f_y), where

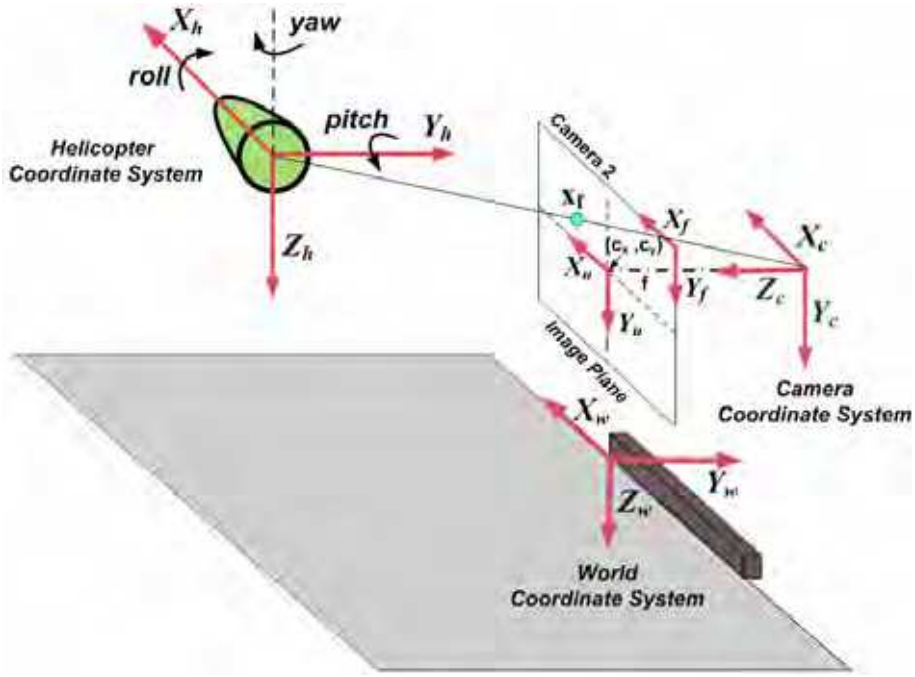

Fig. 3. Coordinate systems involved in the pose estimation algorithm.

$f_x = fm_x$ and $f_y = fm_y$ represent the focal length in terms of pixel dimensions, being m_x and m_y the number of pixels per unit distance.

The above-mentioned camera model assumes that the world point, the image point, and the optical center are collinear; however, in a real camera lens there are some effects (lens distortions) that have to be compensated in order to have a complete model. This compensation can be achieved by the calculation of the distortion coefficients through a calibration process (Zhang (2000)), in which the intrinsic camera parameters, as well as the radial and tangential distortion coefficients, are calculated.

- ***Camera* and *World Coordinate Systems***

Considering that the cameras are fixed, these systems are related by a rigid transformation that allows to define the pose of the k^{th} camera in a *World Coordinate Frame*. As presented in equation (16), this transformation is defined by a rotation matrix $\mathbf{R^k}$ and a translation vector $\mathbf{t^k}$ that link the two coordinate systems and represent the extrinsic camera parameters. Such parameters are calculated through a calibration process of the trinocular system.

$$\mathbf{x_c} = \begin{bmatrix} \mathbf{R^k} & \mathbf{t^k} \\ \mathbf{0^T} & 1 \end{bmatrix} \mathbf{x_w} \tag{16}$$

- ***World* and *Helicopter Coordinate Systems***

The *Helicopter Reference System*, as described in figure 3, has its origin at the center of mass of the vehicle and its correspondent axes: X_h, aligned with the helicopter's longitudinal axis; Y_h, transversal to the helicopter; and Z_h, pointing down. Considering that the estimation of the helicopter's pose with respect to the *World Coordinate System* is based on the distribution

of the landmarks around the *Helicopter Coordinate System*, and that the information extracted from the vision system will be used as reference to the flight controller, a relation between those coordinate systems has to be found.
In figure 3, it is possible to observe that this relation depends on a translation vector that defines the helicopter's position ($\mathbf{t}$), and on a rotation matrix $\mathbf{R}$ that defines the orientation of the helicopter (*pitch*, *roll* and *yaw* angles). Considering that the helicopter is flying at low velocities ($< 4m/s$), *pitch* and *roll* angles are considered ≈ 0, and only the *yaw* angle (θ) is taken into account in order to send the adequate commands to the helicopter.
Therefore, the relation of the *World* and the *Helicopter Coordinate Systems* can be expressed as follows:

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 & t_x \\ sin(\theta) & cos(\theta) & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_h \\ y_h \\ z_h \\ 1 \end{bmatrix} \tag{17}$$

Where (t_x, t_y, t_z) will represent the position of the helicopter $(x_{w_{uav}}, y_{w_{uav}}, z_{w_{uav}})$ with respect to the *World Coordinate System*, and θ the helicopter's orientation.

3.1 Feature extraction

The backprojection algorithm proposed by *Swain* and *Ballar* in (Swain & Ballard (1991)) is used to extract the different landmarks onboard the UAV. This algorithm finds a *Ratio* histogram Rh_i^k for each landmark i in the k^{th} camera as defined in equation 18:

$$Rh_i^k(j) = min\left[\frac{Mh_i(j)}{Ih^k(j)}, 1\right] \tag{18}$$

This ratio Rh_i^k represents the relation between the bin j of a model histogram Mh_i and the bin j of the histogram of the image Ih^k which is the image of the k_{th} camera that is being analyzed. Once Rh_i^k is found, it is then backprojected onto the image. The resulting image is a gray-scaled image, whose pixel's values represent the probability that each pixel belongs to the color we are looking for.
The location of the landmarks in the different frames are found using the previous-mentioned algorithm and the *Continuously Adaptive Mean Shift* (*CamShift*) algorithm (Bradski (1998)). The *CamShift* takes the probability image for each landmark i in each camera k and moves a search window (previously initialized) iteratively in order to find the densest region (the peak) which will correspond to the object of interest (colored-landmark i). The centroid of each landmarks ($\bar{x}_i^k$, $\bar{y}_i^k$) is determined using the information contained inside the search window to calculate the zeroth ($m_{i_{00}}^k$), and first order moments ($m_{i_{10}}^k$, $m_{i_{01}}^k$), (equation 19). These centroids found in the different images (as presented in figure. 4) are then used as features for the 3D reconstruction stage.

$$\bar{x}_i^k = \frac{m_{i_{10}}^k}{m_{i_{00}}^k}; \quad \bar{y}_i^k = \frac{m_{i_{01}}^k}{m_{i_{00}}^k} \tag{19}$$

When working with overlapping FOVs in a 3D reconstruction process, it is necessary to find the relation of the information between the different cameras. This process is known as

Fig. 4. Feature Extraction. Different features must be extracted from images taken by different cameras. In this example color-based features have been considered.

feature matching. This is a critical process, which requires the differentiation of features in the same image and also the definition of a metric which tells us if the feature *i* in image $\mathbf{I^1}$ is the same feature *i* in image $\mathbf{I^2}$ (image -**I**- of camera *k*).
However, in this case, the feature matching problem has been solved taking into account the color information of the different landmarks; so that, for each image $\mathbf{I^k}$ there is a matrix $\mathbf{F^k_{4\times 2}}$ that will contain the coordinates of the features *i* found in this image. Then, the features are matched by grouping only the characteristics found (the central moments of each landmark) with the same color, that will correspond to the information of the cameras that are seing the same landmarks.

3.1.1 3D reconstruction

Assuming that the intrinsic parameters ($\mathbf{K^k}$) and the extrinsic parameters ($\mathbf{R^k}$ and $\mathbf{t^k}$) of each camera are known (calculated through a calibration process), the 3D position of the matched landmarks can be recovered by intersecting in the 3D space the backprojection of the rays from the different cameras that represent the same landmark.
The relation of the found position of each landmark, expressed in the *Lateral Coordinate System* (image plane), with the position expressed in the *Camera Coordinate System*, is defined as:

$$x^k_{f_i} - c^k_x = f^k_x \frac{x^k_{c_i}}{z^k_{c_i}}, \quad y^k_{f_i} - c^k_y = f^k_y \frac{y^k_{c_i}}{z^k_{c_i}} \tag{20}$$

where ($x^k_{f_i}$, $y^k_{f_i}$) is the found position of each landmark expressed in the image plane, ($x^k_{c_i}$, $y^k_{c_i}$, $z^k_{c_i}$) represent the coordinates of the landmark expressed in the *Camera Coordinate System*, (c^k_x , c^k_y) the coordinates of the center of projection in pixel units, and (f^k_x , f^k_y) the focal length in terms of pixel dimensions.
If the relation of the 3D position of landmark i with its projection in each *Camera Coordinate System* is defined as:

$$\mathbf{x^k_{c_i}} = \begin{bmatrix} \mathbf{R^k} & \mathbf{t^k} \\ \mathbf{0^T} & 1 \end{bmatrix} \mathbf{x_{w_i}} \tag{21}$$

Then, integrating equation 21 and equation 20, and reorganizing them, it is possible to obtain the following equations:

$$x_{u_i}^k = f^k \frac{r_{11}^k x_{w_i} + r_{12}^k y_{w_i} + r_{13}^k z_{w_i} + t_x^k}{r_{31}^k x_{w_i} + r_{32}^k y_{w_i} + r_{33}^k z_{w_i} + t_z^k} \tag{22}$$

$$y_{u_i}^k = f^k \frac{r_{21}^k x_{w_i} + r_{22}^k y_{w_i} + r_{23}^k z_{w_i} + t_y^k}{r_{31}^k x_{w_i} + r_{32}^k y_{w_i} + r_{33}^k z_{w_i} + t_z^k} \tag{23}$$

Where $x_{u_i}^k$ and $y_{u_i}^k$ represent the coordinates of landmark *i* expressed in the *Central Camera Coordinate System* of the k^{th} camera, r^k and t^k are the components of the rotation matrix $\mathbf{R^k}$ and the translation vector $\mathbf{t^k}$ that represent the extrinsic parameters, and x_{w_i}, y_{w_i}, z_{w_i} are the 3D coordinates of landmark *i*.

From equations 22 and 23 we have a linear system of two equations and three unknowns with the following form:

$$\begin{aligned}(x_{c_i}^1 r_{31}^1 - r_{11}^1)x_{w_i} + (x_{c_i}^1 r_{32}^1 - r_{12}^1)y_{w_i} + (x_{c_i}^1 r_{33}^1 - r_{13}^1)z_{w_i} = \\ (t_x^1 - x_{c_i}^k t_z^1) \\ (y_{c_i}^1 r_{31}^1 - r_{21}^1)x_{w_i} + (y_{c_i}^1 r_{32}^1 - r_{22}^1)y_{w_i} + (y_{c_i}^1 r_{33}^1 - r_{23}^1)z_{w_i} = \\ (t_y^1 - y_{c_i}^k t_z^1)\end{aligned} \tag{24}$$

$$\mathbf{Ac} = \mathbf{b}$$

If there are at least two cameras seeing the same landmark, it is possible to solve the overdetermined system using the least squares method whose solution will be equation 25, where the obtained vector $\mathbf{c}$ represents the 3D position (x_{w_i}, y_{w_i}, z_{w_i}) of the i^{th} landmark:

$$\mathbf{c} \approx \mathbf{A}^{+}\mathbf{b} = (\mathbf{A}^{\mathbf{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathbf{T}}\mathbf{b} \tag{25}$$

Once the 3D coordinates of the landmarks onboard the UAV have been calculated, the UAV's position ($\mathbf{x}_{\mathbf{w}_{\mathbf{uav}}}$) and its orientation with respect to *World Coordinate System* can be estimated using the 3D position found and the landmark's distribution around the *Helicopter Coordinate System* (see figure 5). The helicopter's orientation is defined only with respect to the Z_h axis (Yaw angle θ) and it is assumed that the angles, with respect to the other axes, are considered to be ≈ 0 (helicopter on hover state or flying at low velocities < 4 m/s). Therefore, equation 17 can be formulated for each landmark.

Reorganizing equation 17, considering that $c\theta = \cos(\theta)$, $s\theta = \sin(\theta)$, $x_{w_{uav}} = t_x$, $y_{w_{uav}} = t_y$, $z_{w_{uav}} = t_z$, and formulating equation 17 for all the landmarks detected, it is possible to create a system of equations of the form $\mathbf{Ac} = \mathbf{b}$ as in equation 26, with five unknowns: $c\theta$, $s\theta$, $x_{w_{uav}}$, $y_{w_{uav}}$, $z_{w_{uav}}$. If at least the 3D position of two landmarks is known, this system of equations can be solved as in equation 25, and the solution $\mathbf{c}$ is a 4 × 1 vector whose components define the orientation (*yaw* angle) and the position of the helicopter expressed with respect to a *World Coordinate System*.

$$\begin{bmatrix} x_{h_1} & -y_{h_1} & 1 & 0 & 0 \\ y_{h_1} & x_{h_1} & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{h_i} & -y_{h_i} & 1 & 0 & 0 \\ y_{h_i} & x_{h_i} & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta \\ s\theta \\ x_{w_{uav}} \\ y_{w_{uav}} \\ z_{w_{uav}} \end{bmatrix} = \begin{bmatrix} x_{w_1} \\ y_{w_1} \\ z_{w_1} - z_{h_1} \\ \vdots \\ x_{w_i} \\ y_{w_i} \\ z_{w_i} - z_{h_i} \end{bmatrix} \tag{26}$$

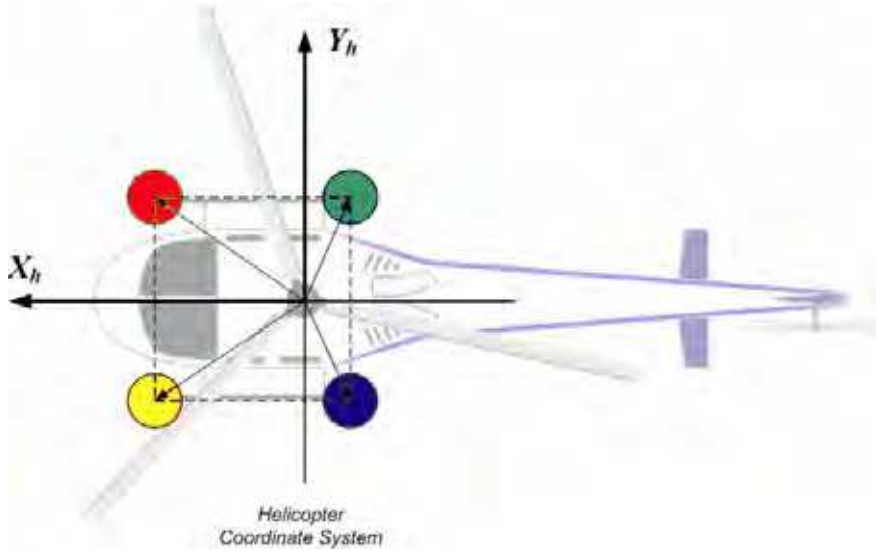

Fig. 5. Distribution of landmarks. The distribution of the landmarks in the *Helicopter coordinate system* is a known parameter used to extract the helicopter position and orientation with respect to the *World coordinate system*.

In figures: 6(a), 6(b), 6(c) and 6(d), it is possible to see an example of the UAV's position estimation using a ground-based multi camera system (see Martínez et al. (2009) for more details). In these figures, the vision-based position and orientation estimation (red lines) is also compared with the estimation obtained by the onboard sensors of the UAV (green lines).

4. Onboard visual system for pose estimation

In this section, a 3D pose estimation method based on projection matrix and homographies is explained. The method estimates the position of a world plane relative to the camera projection center for every image sequence using previous frame-to-frame homographies and the projective transformation at first frame, obtaining for each new image, the camera rotation matrix $\mathbf{R}$ and a translational vector $\mathbf{t}$. This method is based on the propose by Simon *et. al.* (Simon et al. (2000), Simon & Berger (2002)).

4.1 World plane projection onto the Image plane

In order to align the planar object on the world space and the camera axis system, we consider the general pinhole camera model and the homogeneous camera projection matrix, that maps a world point $\mathbf{x}_w$ in $\mathbb{P}^3$ (projective space) to a point $\mathbf{x}^i$ on i^{th} image in $\mathbb{P}^2$, defined by equation 27:

$$s\mathbf{x}^i = \mathbf{P}^i\mathbf{x}_w = \mathbf{K}[\mathbf{R}^i|\mathbf{t}^i]\mathbf{x}_w = \mathbf{K}\begin{bmatrix}\mathbf{r}_1^i & \mathbf{r}_2^i & \mathbf{r}_3^i & \mathbf{t}^i\end{bmatrix}\mathbf{x}_w \tag{27}$$

where the matrix $\mathbf{K}$ is the camera calibration matrix, $\mathbf{R}^i$ and $\mathbf{t}^i$ are the rotation and translation that relates the world coordinate system and camera coordinate system, and s is an arbitrary

scale factor. Figure 7 shows the relation between a world reference plane and two images taken by a moving camera, showing the homography induced by a plane between these two frames.

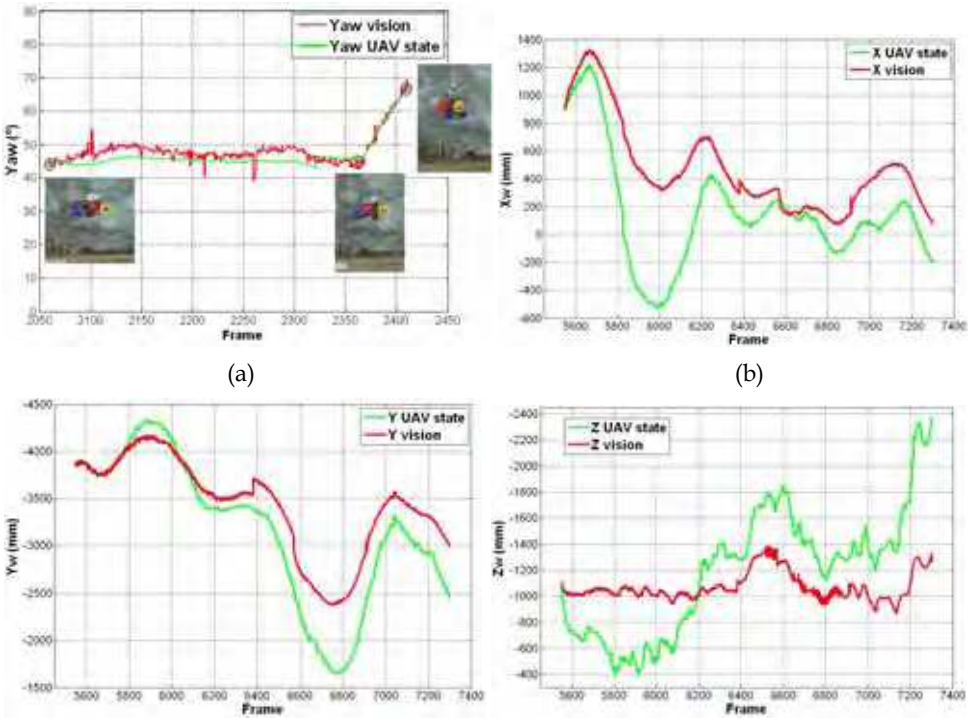

Fig. 6. Vision-based estimation vs. helicopter state estimation. The state values given by the helicopter state estimator after a *Kalman filter* (green lines) are compared with a multiple view-based estimation of the helicopter's pose (red lines).

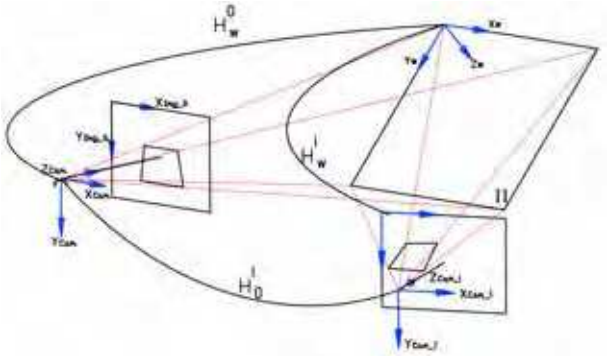

Fig. 7. Projection model on a moving camera and frame-to-frame homography induced by a plane.