Open Archive Toulouse Archive Ouverte (OATAO)

OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible.

This is an author-deposited version published in: http://oatao.univ-toulouse.fr/
Eprints ID: 8284


Any correspondence concerning this service should be sent to the repository administrator: staff-oatao@inp-toulouse.fr
Stabilization of an Unmanned Aerial Vehicle Using Real-Time Embedded Motion Estimation

Matthieu Claybrough1 and François Defay2

Abstract—This paper presents an image-based velocity control method for a VTOL UAV. For increased autonomy, a real-time embedded global motion estimation process using efficient feature tracking was developed. The VTOL vehicle is assumed to be equipped with a minimum sensor suite—i.e., a camera, altitude sensor and IMU—and to hover over a featured flat plane. This paper first presents a fast feature-based global motion estimation method, and then proposes a control strategy based on this method. Finally, experimental results using a quadrotor demonstrate the performance of the proposed control strategy and feature-based motion estimation.

Index Terms—Feature extraction, motion estimation, real time systems, unmanned aerial vehicle (UAV).

I. INTRODUCTION

Recent advances in technology have led to computer vision gaining importance as a low-cost but information-rich source for complementing the sensor suite of Unmanned Aerial Vehicles (UAVs). Common sensor suites generally include an Inertial Measurement Unit (IMU), a Global Positioning System (GPS), and a sonar [1], [2]. This suite is sufficient for stabilizing the UAV’s attitude [3], but due to both IMU and GPS limitations, particularly in indoor flight, position estimation and control become difficult. Vision-based state estimation and control are currently active research topics. This twofold task is complex because it involves hardware integration, computationally intensive low-level image processing, multiple-view geometry for movement extraction, and finally the synthesis of real-time controllers.

To limit hardware integration and the constraints of real-time onboard processing times, many positive results have been obtained by sending the images to a ground station for processing. In [4], a method is proposed for landing a vertical take-off and landing (VTOL) UAV on a flat textured plane. The motion is estimated by a ground station using a pyramidal implementation of the Lucas-Kanade algorithm [5]. In [6] and [7], the processing and segmentation tasks are performed by a ground station to extract features from a highly contrasted landing pad situated beneath the UAV. Both propose real-time stabilization and position tracking using pose reconstruction.

Aiming for greater UAV autonomy, this paper presents a real-time, computationally cheap method for estimating global motion. More precisely, the feature-based tracking system only requires local low-level image processing and is able to operate on any sufficiently featured flat plane. A method for deriving absolute UAV velocity and a controller for stabilizing the UAV is proposed. Experimental results validate the effectiveness of the method.

II. GLOBAL MOTION ESTIMATION

Global motion estimation methods can be classified into two main types. Direct methods estimate motion directly from the brightness of each pixel and assume that brightness does not differ from one image to the next [8]. These methods include differential methods, phase correlation and block-matching algorithms. The performance of optical flow methods is discussed in [9]. They have in common the need to process many if not all pixels to estimate motion. Pyramidal [5] and parallel implementations [10] have been proposed to decrease processing time but it remains long.

A feature-based motion estimation system was chosen to limit the computational complexity. Use of such a method can drastically reduce the calculation time by limiting the low-level image processing to a few carefully selected zones. Recent research in feature extraction methods has furthermore led to fast extraction algorithms. The following sections explain the implementation in further detail.

A. Feature Choice and Extraction

For a feature-based estimation method, the first step is always entails extracting and classifying features. There is a variety of good feature detectors such as Harris [11], [12] and SUSAN [13]. For our real-time application, the main criteria for selecting the detector were repeatability and computational simplicity. Various detectors were benchmarked and the FAST detector [14] outperformed all others on the computational criterion whilst offering good repeatability. See section IV for experimental results.

The main drawback to the FAST detector is sensitivity to noise and scale. Though noise can be filtered, the scale issue has to be fully mastered as our application involves quick variations in height and therefore in image scale, as will be seen in the motion estimation equations.
B. Increasing Tracking Speed

To minimize the overall computational complexity of the motion estimation system, it is obviously necessary to minimize each step’s complexity. As seen above, the task can be divided into three steps:

- **Step 1: Feature extraction**
- **Step 2: Feature matching**
- **Step 3: Motion extraction**

For a given algorithm, FAST in our case, the first step’s complexity is proportional to the number of pixels that are tested as candidates to be features. The algorithm also requires a few more operations for each extracted feature. The complexity of the second and third steps is directly linked to the number of features being tracked. The two ways to increase tracking speed are therefore to reduce the image surface to be processed by the feature extraction algorithm and to reduce the number of features being tracked.

The optimal number of points to track is therefore the minimum number that allows good motion estimation. Assuming that image motion is described by a 4-parameter homographic mapping, at least two points are required to unambiguously determine the homography [15]. Due to outliers in the match data and to obtain a robust estimation, it is necessary to increase the number of points. Similarly, some features will not be found in each frame, and it is therefore necessary to keep a margin for reliable and continuous motion estimation. The optimal number of points is determined by the motion extraction method.

The main idea for reducing the image surface that has to be processed is to only process those zones where a feature is expected. The tracking system developed keeps a database of the extracted features with their location, history and strength. A model is used to predict the next position of all the features along with the uncertainty and predicted noise. A search zone is then defined for each tracked point, centered on the predicted position. Its size depends on the predicted uncertainty and noise. The total surface to be processed is therefore the sum of each tracked point’s search zone. To minimize this surface, the number of tracked features must once again be limited to a minimum and the size of each search zone reduced. The trade-off when reducing search zone size is increasing the probability of losing tracked features if they are outside the zone: the more accurate the prediction, the smaller the search zones and the faster the feature extraction.

The second step usually involves matching all the extracted features from one image to the next. In our case, this task is simplified by the fact that features were extracted in order to find known tracked features. The matching step is used to validate the matches and resolve uncertainties when multiple features are found in the same search zone for example.

For our tracking system to perform well, it is necessary to add new features when there are not enough tracked features in the database. This occurs when the algorithm is initialized and when the scenery beneath the UAV changes. To do this, an exploration routine whose effort is linked to the number of required new points was developed. To meet real-time constraints, exploration is spread over successive images and uses a heuristic approach to choose the exploration zones. For this purpose, previous explorations are kept in memory and the effort is intensified in unexplored zones or that have remained unexplored for a long time. The predicted movement is also used to locate the unexplored zones. Finally, exploration of zones containing tracked features are avoided to spread out the tracked points over the entire scene.

C. Motion Estimation Equations

1) Frames and Transformation Matrix: The VTOL UAV is represented by a rigid body of mass \( m \) whose centre of gravity is denoted \( G \). The position of the vehicle expressed in the inertial reference frame \( I = (X_0, Y_0, Z_0) \) with \( Z_0 \) pointing upward is \( \xi = (x, y, z)^{T} \). The orientation of the vehicle is represented by the three Euler angles (roll, pitch and yaw) and expressed as \( \eta = (\phi, \theta, \psi)^{T} \), also relative to the inertial frame. Let \( (X_b, Y_b, Z_b) \) represent the body fixed frame, with \( Z_b \) representing the yaw axis and pointing upwards, and with \( X_b \) and \( Y_b \) the pitch and roll axes.

Let \( M_b \) represent the transformation matrix from the inertial reference frame to the UAV fixed body frame, thus:

\[
M_b = \begin{pmatrix}
R_{\phi, \theta, \psi} & 0 \\
0 & 0 \\
0 & 0 & 1
\end{pmatrix}
\]

where \( R_{\phi, \theta, \psi} \) is the Euler rotation matrix with the ZYX rotation order.
Finally, let \( M_c \) represent the world transformation matrix from the fixed body frame to the camera frame.

2) Camera Matrix Model: To model the camera projection a \( 3 \times 4 \) camera projection matrix \( C \) was used. This matrix can be identified offline. For a quick identification, the pinhole camera model was used, leading to a simplified projection matrix:

\[
C = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & \frac{1}{f} & 0
\end{pmatrix}
\]

where \( f \) is the camera’s focal length.

Let \( \mathbf{p} = (x_p, y_p, z_p, 1) \) represent the homogeneous coordinates of a tracked feature on the ground, expressed in the inertial frame \( I \) and \( p' = (x'_p, y'_p, z'_p) \) the homogeneous coordinates representing the projection of the tracked feature on the camera image. Letting \( \xi = (x, y, z, 0) \) yields:

\[
p' = CM_cM_b(\mathbf{p} - \xi).
\] (1)

3) Explicit Motion Estimation Formulas: Transforming (1) back to centered pixel coordinates \((\hat{x}_p, \hat{y}_p)\) yields:

\[
\hat{x}_p = \frac{x_p}{z_p}
\] (2)

\[
\hat{y}_p = \frac{y_p}{z_p}
\] (3)

To limit calculation complexity, the following assumptions are made:

- All tracked features are in the horizontal plane, i.e. \( z_p = 0 \).
- The pitch and roll angles are small enough to use first order Taylor series for cosine and sine, i.e. \( |\phi| < \epsilon_{\phi\theta} \) and \( |\theta| < \epsilon_{\phi\theta} \) with \( \epsilon_{\phi\theta} \) a small parameter.
- The camera is mounted facing downwards and its centre coincides with the UAV’s centre of gravity.

Under these assumptions (2) and (3) can be rewritten as:

\[
\hat{x}_p = f \frac{c_\phi(x_p - x) + s_\phi(y_p - y) - \theta z}{(\phi s_\theta - \theta c_\theta)(x_p - x) + (\phi c_\theta + \theta s_\theta)(y_p - y) - z}
\] (4)

\[
\hat{y}_p = f \frac{s_\phi(x_p - x) - c_\phi(y_p - y) + \phi z}{(\phi s_\theta - \theta c_\theta)(x_p - x) + (\phi c_\theta + \theta s_\theta)(y_p - y) - z}
\] (5)

Furthermore, the camera’s limited angle of view along with the assumption of small pitch and roll angles ensures that the following constraints hold with \( \beta \approx 1 \):

\[
|x_p - x| < \beta z \quad \text{and} \quad |y_p - y| < \beta z
\] (6)

Using the fact that the pitch and roll angles are smaller than \( \epsilon_{\phi\theta} \) gives:

\[
|\phi s_\theta - \theta c_\theta|(x_p - x) < \sqrt{2}\epsilon_{\phi\theta}\beta z
\] (7)

\[
|\phi c_\theta + \theta s_\theta|(y_p - y) < \sqrt{2}\epsilon_{\phi\theta}\beta z
\] (8)

The denominator of the previous equations can therefore be simplified, which leads to:

\[
\hat{x}_p = \frac{-f}{z}(c_\phi(x_p - x) + s_\phi(y_p - y) - \theta z)
\] (9)

\[
\hat{y}_p = \frac{-f}{z}(s_\phi(x_p - x) - c_\phi(y_p - y) + \phi z)
\] (10)

To extract the optical flow equations the time derivative of both expressions is calculated. Without loss of generality, \( x = y = \psi = 0 \) is assumed to simplify the demonstration. This is possible by simply rewriting the equations in a new reference frame, obtained by a \((-x, -y, 0)^T\) translation followed by a \( \psi \) rotation around the \( Z \) axis.

\[
\hat{x}'_p = \frac{zf}{z^2}(x_p - \theta z) - \frac{f}{z}(\psi y_p - \dot{x} - \dot{\theta} z - \theta \dot{z})
\] (11)

\[
\hat{y}'_p = \frac{zf}{z^2}(-y_p + \phi z) - \frac{f}{z}(\psi x_p + \ddot{\psi} + \phi + \dot{\phi} z)
\] (12)

Considering two tracked points and subtracting their coordinates further simplifies the equations. More precisely, let us introduce the following notations:

\[
\Delta \hat{x} = \hat{x}_{p_2} - \hat{x}_{p_1}, \quad \Delta \hat{y} = \hat{y}_{p_2} - \hat{y}_{p_1}
\] (13)

\[
\Delta \hat{x}' = \hat{x}'_{p_2} - \hat{x}'_{p_1}, \quad \Delta \hat{y}' = \hat{y}'_{p_2} - \hat{y}'_{p_1}
\] (14)

Direct application of equations (11) and (12) gives the 4 following equations, linking \( \psi \) and \( \dot{\psi} \) to the coordinates of extracted points:

\[
\frac{\Delta \hat{x}}{\Delta \hat{y}} = -\psi - \Delta \hat{x} \ddot{\psi} \quad \frac{\Delta \hat{y}}{\Delta \hat{x}} = -\dot{\psi} + \Delta \hat{x} \ddot{\psi}
\] (15)

\[
\frac{\Delta \hat{y}}{\Delta \hat{x}} = -\psi - \Delta \hat{y} \ddot{\psi} \quad \frac{\Delta \hat{x}}{\Delta \hat{y}} = -\dot{\psi} - \Delta \hat{y} \ddot{\psi}
\] (16)

Finally, these equations can combined two-by-two to obtain an explicit formula for \( \ddot{\psi} \) and \( \dddot{\psi} \) using only two tracked points:

\[
\dddot{\psi} = \frac{\Delta \hat{y} \dddot{\hat{y}} + \Delta \hat{x} \dddot{\hat{x}}}{\Delta \hat{x}^2 + \Delta \hat{y}^2}
\] (17)

\[
\dddot{\hat{z}} = \frac{\Delta \hat{y} \dddot{\hat{y}} + \Delta \hat{x} \dddot{\hat{x}}}{\Delta \hat{x}^2 - \Delta \hat{y}^2}
\] (18)

Likewise, explicit formulas for \( \hat{x} \) and \( \hat{y} \) can be obtained by rearranging equations (11) and (12) yields:

\[
\hat{x} = \frac{zf}{z^2}x_{pxl} - \theta z - \theta \dot{z} + z \psi \dot{\phi}
\] (19)

\[
\hat{y} = \frac{zf}{z^2}y_{pxl} - \phi z - \phi \dot{z} - z \psi \dot{\theta}
\] (20)

where

\[
x_{pxl} = +\hat{x}' + \frac{\dot{\psi}}{z} x_p + \psi \dot{y}_p
\] (21)

\[
y_{pxl} = -\hat{y}' + \frac{\dot{\psi}}{z} y_p + \psi \dot{x}_p
\] (22)

represent the optical flow in pixels and can be fully calculated using extracted points. To fully determine \((\hat{x}, \hat{y})\), it is necessary to know \( z \) and \((\phi, \dot{\phi}, \theta, \dot{\theta})\) angles. These parameters can be calculated using a 8-parameter homographic mapping [15] but this is computationally intensive and the onboard IMU already provides a good
estimation of these angles and angular speeds. The altitude $z$ is estimated using a sonar.

The IMU can also provide $\psi$ and $\dot{\psi}$ which allows different data fusion scenarios. One can either merge both IMU and optical flow values, or use one to correct the other. In practise, the optical flow estimated $\dot{\psi}$ proved to be more reliable.

D. Motion Estimation Algorithm

The previous paragraph demonstrates a method for generating $(\psi, \dot{\psi})$ couples using pairs of tracked features and how to estimate a $(x_{pix}, y_{pix})$ vector from each tracked point once $\psi$ and $\dot{\psi}$ are known. Using these explicit formulas a six step process was implemented to extract global motion parameters:

- Combine tracked points two-by-two to generate usable pairs
- Calculate $\dot{\psi}_i$ and $\dot{\dot{z}}_i$ for each pair
- Use a statistic algorithm for estimating $\psi$ and $\dot{\psi}$
- Calculate $(x_{pix}, y_{pix})_i$, vector for each extracted point
- Use a statistic algorithm for estimating $(x_{pix}, y_{pix})$
- Use synchronized IMU and sonar estimations of $(\dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi})$ to calculate $(\ddot{x}, \ddot{\gamma})$

With $n$ representing the number of tracked points, there are $\frac{n(n-1)}{2}$ possible pairs. For quick but robust estimation a a similar number of pairs is needed for each frame, independently of $n$. Furthermore, only pairs such as $|\Delta \dot{x}| > \epsilon$, $|\Delta \dot{y}| > \epsilon$ and $|\Delta \ddot{x}^2 - \Delta \ddot{y}^2| > \epsilon$ are usable. An algorithm meeting both needs was developed: all valid pairs are generated when $n$ is small and as $n$ increases the selectivity increases to keep the number of pairs sufficiently constant. Values for $\dot{\psi}$ and $\dot{\dot{z}}$ are then generated using the exposed explicit formulas (19) and (20).

A non-probabilistic version of the RANSAC [16] algorithm was implemented to estimate the motion parameters. The algorithm starts by considering different possible estimations based on the dataset and then selects similar values from the dataset. If there are enough of them, their mean value is calculated and becomes a possible estimation. The mean square error of each candidate is calculated in order to select the best estimation. This method is particularly robust to outliers in the dataset and much faster than a least-square estimation. Our algorithm is executed three times for each motion estimation: $\dot{\psi}$ and $\dot{\dot{z}}$ are estimated independently whilst $x_{pix}$ and $y_{pix}$ are estimated simultaneously using the Euclidian L2 norm. The algorithm also outputs the number of values and the mean square error. These values are used to form an estimation reliability indicator that is then used by the data fusion process.

III. UNMANNED AERIAL VEHICLE STABILIZATION

A. Dynamic Model

The dynamic model for our UAV can be obtained by representing the quadrotor as a combination of two planar VTOLs, each with an independent torque control. The model can then be derived from the Euler-Lagrange equations as can be seen in [7]. Neglecting the coupling between the rolling and lateral acceleration and considering the small angles assumption yields the following simplified model:

$$m\ddot{x} = (\sin \theta)u$$

$$m\ddot{y} = (\sin \phi)u$$

$$m\ddot{z} = (\cos \theta \cos \phi)u - mg$$

$$\ddot{\psi} = \tilde{\tau}_\psi$$

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

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

where $u$ is the main thrust and $g$ the gravitational constant. $\tilde{\tau}_\psi$, $\tilde{\tau}_\theta$ and $\tilde{\tau}_\phi$ are respectively the yawing, pitching and rolling moments, which are related to the generalized torques $\tau_\psi$, $\tau_\theta$ and $\tau_\phi$.

B. Stabilization Control Law

For stabilizing the quadrotor, two dynamics can be distinguished: attitude stability and translational stability. A common architecture uses a cascaded loop structure in which attitude is stabilized in the inner-loop and position in the outer-loop. This is possible under the key assumption that both sets of dynamics are time-scale separated. Attitude and altitude are stabilized in the inner-loop by the low-level control board using 4 independant PID controllers whilst the horizontal speed and position is controlled by issuing desired pitch and roll angles in the outer-loop . The final pitch and roll controllers result in a second order transfer function:

$$H(p) = \frac{1}{1 + \frac{2\xi}{\omega_0}p + \left(\frac{p}{\omega_0}\right)^2}$$

with $\xi = 0.7$ and $\omega_0 = 6$ rad/s resulting in a 0.4s response time. Similar transfer functions for altitude and yaw control enable stable and well-damped responses for all three angles and attitude.

A PI controller for horizontal velocity was implemented in the outer-loop. Setting the desired speed to 0 thus corresponds to a position hold. The speed controller also makes piloting more accessible and enables use of high-end mission planners. The PI controller was implemented as follows:

$$\phi_d = -k(\dot{x} - \dot{x}_d) - k_i \int (\dot{x} - \dot{x}_d)$$

$$\theta_d = -k(\dot{y} - \dot{y}_d) - k_i \int (\dot{y} - \dot{y}_d)$$

where $\dot{x}_d$ and $\dot{y}_d$ represent the desired horizontal speeds and $k$ and $k_i$ are controller parameters.
IV. EXPERIMENTAL RESULTS

A. Prototype Description

The proposed vision algorithm was tested on board a quadrotor UAV developed at ISAE, France. The onboard electronics consist of two connected boards: a low-level board containing the IMU and controlling the motor drivers, and a high-level processing unit running the vision algorithms and handling communication with the ground station and low-level board. The low-level board ensures angular stability and can land the UAV if the high-level processing unit fails. The main properties can be summarized as follows:

- **Quadrotor**: 850g, 55cm between the rotor tips, developed at DMIA, ISAE, France.
- **Low-level board**: Also developed in-house, the low-level board combines an ITG-3200 gyro, ADXL345 accelerometer and an HMC-5843 magnetometer.
- **Camera**: Gumstix Caspa VL: Global shutter CMOS, 752x480px, color, max 60fps.
- **High level board**: Gumstix Overo: A computer-on-module delivering an ARM Cortex-A8 processor at 720MHz.

Both the vision algorithms and control methods were run in real time on board the high-level board. A WiFi-based datalink allows a real-time view of high-level algorithm information, such as the position and number of tracked points, motion estimation filter status, etc. The datalink can also be used to upload new parameters on the fly.

B. Experiments & Results

Various detectors were benchmarked to select the corner extraction algorithm. The following results are the average execution time over 1000 iterations of a 320x240 8 bit grayscale video sequence. The sensitivity of each detector was calibrated to extract approximately 60 corners per frame. The test PC contains an Intel I7 processor and the embedded calculator is the 720MHz Gumstix board.

For the vision system to perform well over various surfaces, the FAST detector’s sensitivity is automatically adjusted in order to extract a sufficient number of features – corners in the case of the FAST algorithm. Similarly, automatic setting of the camera parameters ensures robustness against changes in lighting.

### TABLE I

<table>
<thead>
<tr>
<th>Corner detector</th>
<th>Harris</th>
<th>Min-Eigen</th>
<th>FAST</th>
</tr>
</thead>
<tbody>
<tr>
<td>17 PC (ms)</td>
<td>3.2</td>
<td>2.7</td>
<td>0.41</td>
</tr>
<tr>
<td>Gumstix (ms)</td>
<td>405</td>
<td>504</td>
<td>4.3</td>
</tr>
</tbody>
</table>

The experimental results that follow were obtained over an office floor with a few wear marks or stains. The system was tested between an altitude range of 0.5 to 2m and proved capable of automatic landing. Noise was filtered using the RANSAC algorithm to remove incorrect matches and a low-pass filter for the horizontal velocities.

To evaluate the speed and efficiency of the tracking system, many status variables are logged during each flight. As explained previously, key parameters include the image surface processed, the number of tracked points and the success rate in tracking the points. As can be seen in Fig. 3, the tracking system performs well with minimal computational impact: approximately 5% of the image is processed for tracking and 3% for exploration. Furthermore, the total computational time remains under 2ms. These results were obtained with low horizontal speeds, but experiments show that the system keeps track even at high speeds, as long as the altitude does not vary too quickly.

To evaluate the motion estimation precision, the quadrotor was placed on a test bench allowing only the three angular movements and performed the following acquisitions. Fig. 4 plots the estimated optical pixel-speed \((x_{\text{opt}}, y_{\text{opt}})\) and the reference speed obtained by multiplying the angular rates from the gyroscopes by the fixed altitude. The video frequency and sampling rate was 20Hz. The root mean...
square error between the filtered estimate and the reference is approximately 10pxl/s, i.e. less than 4mm/s when hovering one metre from the ground.

Fig. 4. Estimated flow, filtered estimate and reference

Fig. 5 proves the efficiency of the proposed horizontal velocity stabilization law. The controller settings were $k = 10$ and $k_i = 0.15$. The left plots demonstrate a position hold, i.e. $\dot{x}_d = \dot{y}_d = 0$ whilst the right plots show the impulse response function. The video acquisition was set at 25Hz but the control loop and sampling runs at 100Hz, thus each variable is plotted over a 10-second time span. Fig. 5 only plots the data concerning the X-axis as similar results are obtained for the Y-axis.

V. CONCLUSIONS AND FUTURE WORK

A feature-based global motion estimation method was proposed to efficiently estimate a quadrotor UAV’s translational velocity by processing less than 10% of the image, with an embedded execution time below 2ms. A full state feedback control was reviewed and applied allowing for real-time autonomous velocity stabilization. The velocity controller outputs the desired roll and pitch angles and a higher frequency controller ensures these angles are obtained. Experimental results demonstrate the good performance of this motion estimation algorithm and control strategy.

Future work will focus on two main tasks. The first entails improving the velocity response and testing the algorithm at higher translational speeds and outdoors. The second consists in refining the feature selection method and automatic camera settings to ensure efficient motion estimation on many different surfaces, both indoors and outdoors. Finally, the data fusion algorithms and models need to be improved to ensure velocity stability for the short periods when the vision-based motion estimation fails. This improved autonomy is particularly useful for crossing non-featured zones.

REFERENCES