Optic-Flow based estimation of angle of attack

Semester Project

Brice Platerrier

Section Microtechnique

FEBRUARY 2015 / 2016

Assistants:
Julien Lecoeur

Laboratory of Intelligent Systems (LIS)
Prof. Dario Floreano

Abstract

The Ywing is a VTOL (vertical take off and landing) drone capable of flying at any speed between its cruise speed and hover by varying its angle of attack. The angle of attack (angle between wing chord and airflow) cannot always be approximated by pitch angle (angle between wing chord and horizon) because the drone is not always flying at constant altitude or undergoes some wind. The goal of this project was to use optic flow measurements from a wide field of view camera embedded on the drone to estimate the angle of attack. Indeed, in translation flight, all optic flow vectors point to the current direction of flight, which can be used to estimate angle of attack. The estimation of angle of attack was implemented on the onboard controller using optic flow vectors computed on the camera, and validated using motion tracking systems.

Contents

1   Introduction

Monocular vision sensors are probably the most widely used devices in embedded systems, where the payload is a serious parameter to take into consideration, even more so for flying robots. Depending on the application, whether it be obstacle avoidance or egomotion, several such devices may be used on the embedded system.

Previous work from A. Briod [1], show that the egomotion estimates greatly benefit (statistically) from the addition of numerous pinhole cameras on the body of the robot. Nevertheless, this method entails a calibration procedure to estimate the viewing direction of each sensor with respect to the body of the robot, for which an automatic estimation thereof was proposed in [2]. Hence, another solution is to consider using omnidirectional cameras.

An omnidirectional camera provides a wide field of view of at least 180 degrees. In our case of interest, the dioptric camera that was used consists in a combination of shaped lenses (fisheye lenses) and typically can reach a field of view slightly larger than 180 degrees. Thanks to the calibration of a model of the camera, mapping each camera image pixels to the 3D scene, these devices enable the sampling of many points on the camera image without the need to re-estimate the corresponding viewing directions each time they are changed. More importantly, a wide field-of-view is preferred for optic-flow-based egomotion estimation because it is necessary to distinguish clearly the direction of motion out of multiple optic-flow measurements. It thus allows for more robustness and redundancy in a variety of environments and conditions.

The optical flow (or optic-flow) is the apparent motion of objects, surfaces and edges in a visual scene and can be obtained can be obtained either from the variation of pixel intensity (Lucas-Kanade method) or pattern displacement. This approach has two advantages compared to feature-based techniques. (1) The optic-flow can be estimated from low-resolution sensors, which are typically small and lightweight, allowing for a low computational overhead. (2) The optic-flow measurements can be measured from any scene whether recognizable features are present or not, which enables its use in both indoor and outdoor environments.

But the use of optic-flow measurements has several drawbacks compared to feature tracking when it comes to ego-motion estimation. (1) The optic-flow is caused by the relative motion between the camera and the scene from which some visual cues are extracted. This makes the measurements particularly sensible to outliers induced by large uniform textures (e.g. white walls) or moving objects in the environment. (2) The visual cues which are used for optic-flow computation vary at each timestep causing the distance to the scene to change. Thus, the estimation of the correct scaling of the velocity of the drone may be difficult and typically requires additional sensors such as inertial measurement units [1] or sonar sensors.

Several approches were proposed for inferring the direction of motion from optic-flow measurements. A first approach is to rely on coarse epipolar constraints to estimate the position of the focus of expansion, which is the single point from which the optic-flow field diverges. In this case, the sampling of a high number of optic-flow measurements

at each timestep and the computation of their corresponding epipolar constraints enable the refinement of the estimated position of the FOE [3]. Another approach would be to proceed to optic-flow and an inertial sensor fusion [1]. Relying on translational optic-flow direction constraint and derotation [4], this method enable the estimation of the direction as well as the scaling of the velocity.

However, the above methods exhibit some disadvantages. (1) The computation of epipolar constraints from raw optic-flow measurements is limited to antipodal points, thus reducing the interest in wide field-of-view cameras. (2) The estimation of the scaling factor using extended kalman filters is not necessary in our particular case and requires inertial measurements units which are not present on the camera board (no accelerometers).

In this report, we propose a method for estimating the direction of translation based on high frequency and unscaled optic-flow measurements. We characterize this method by means of a wide field-of-view camera on which a hundred of optic-flow measurements are computed and a motion capture system. Section II describes the theoretical background and the actual algorithm implemented on the camera board. Section III presents the experimental setup used for characterization. Section IV details the results of the experiments conducted in a synthetic environment.

2 Theoretical background

2.1 Omnidirectional Camera Model

All modern fisheye cameras are central, and hence, they satisfy the single effective focal point property. The reason a single effective viewpoint is so desirable is that it allows us to generate geometrically correct perspective images from the pictures captured by the omnidirectional camera. When the geometry of the omnidirectional camera is known, that is, when the camera is calibrated, one can precompute this direction for each pixel. Therefore, each pixel can be mapped onto a plane at any distance from the viewpoint to form a planar perspective image. Additionally, the image can be mapped onto a sphere centered on the single viewpoint, that is, spherical projection.

Omnidirectional camera systems cannot be described using conventional pinhole model because of the very high distortion induced by the imaging device. Indeed, in our case, the model should take into account the refraction caused by the lens of the fisheye camera. A unified model was proposed in [5]. To overcome the lack of knowledge of a parametric model for fisheye cameras, a Taylor polynomial, whose coefficients and degree are found through the calibration process, is used.

Let p be a pixel point of your image, and (u, v) its pixel coordinates with respect to the center of the omnidirectional image. Let P be its corresponding 3D vector emanating from the single effective viewpoint, and (x, y, z) its coordinates with respect to the axis origin. The function estimated by the calibration process maps an image point p into its corresponding 3D vector P:

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} u \\ v \\ f(r) \end{bmatrix} \tag{1}$$

where $\begin{cases} r = \sqrt{u^2 + v^2} \\ f(r) = a_0 + a_1 r + a_2 r^2 + a_3 r^3 + a_4 r^4 + ... \end{cases}$

and the parameters to estimate are a_0, a_1, a_2, ... Although increasing the polynomial may yield better accuracy, we used 4th order polynomials, as a good trade-off between accuracy and complexity of the polynomial model.

However, as the camera and mirror axes are never perfectly aligned, the model is extended so as to model these errors through an affine transformation:

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} c & d \\ e & 1 \end{bmatrix} \cdot \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} x_c \\ y_c \end{bmatrix} \tag{2}$$

which relates the real distorted coordinates (u', v') to the ideal undistorted ones (u, v).

This approximate model of the camera allows for more scalability in the number of optical flow measurements and more flexibility on their location, without the need to proceed to another calibration. Furthermore, the viewing directions corresponding to each pixel could be preprocessed as part of an initialization procedure to speed up subsequent computations.

The proposed calibration procedure relies on the use of a chessboard to automatically locate feature points on the camera images. Unfortunately, the resolution of our camera (160×120) was too low for the edge detection algorithm to perform correctly its task. This issue may be avoided by using circles instead of the squares of the checkerboard. Indeed, this would yield more easily distinguishable corners. Considering this was not part of the current toolbox, we simply located 35 corners manually for each image, prior to calibration. We used 7 images taken from the camera with varying positions and orientation of the chessboard so as to cover most of the its field of view (Fig. 1). In the end, we obtained a subpixel average error of 0.34 pixels and the following model, which relies on a 4th-order Taylor polynomial:

Polynomial: $\begin{cases} a_0 = -6.66.10^1 \\ a_1 = 0.00 \\ a_2 = 6.42.10^{-3} \\ a_3 = -2.31.10^{-5} \\ a_4 = 2.73.10^{-7} \end{cases}$ Center: $\begin{cases} x_c = 56.23 \\ y_c = 77.64 \end{cases}$ Matrix: $\begin{cases} c = 1.00 \\ d = -0.00 \\ e = -0.00 \end{cases}$

As we can see from the coefficients of the matrix in (2), there is not significant misalignment between the camera and the optical lens. Hence, we can neglect this part of the model. Nevertheless, the estimated location of the center is not exactly the actual center of the 160×120 grid of pixels. The corresponding C-code implemented on the camera chip is shown in Code 1.
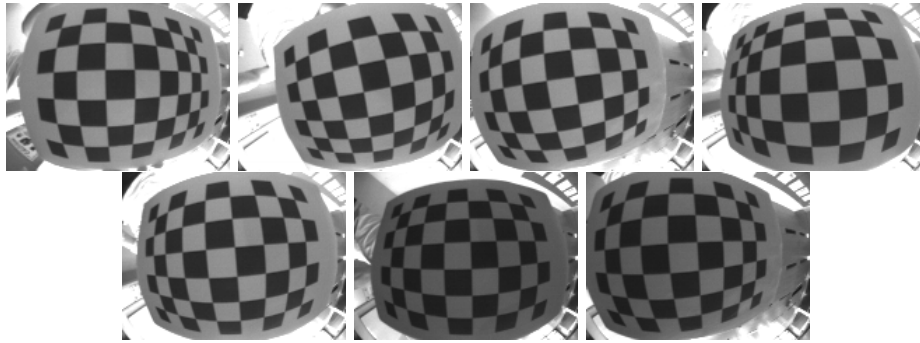
3

Figure 1: Training images used for camera model calibration - The orientation and position of the chessboard were changed from one image to another to most of the field of view. The images are given in grayscale by the camera and exhibit a resolution of 160×120 pixels.

Code 1: From pixels to 3D directions

```c
void cam2world(float *xp, float *yp, float *zp, float u,
        float v, cam_model *cam)
{
    float *pol     = cam->pol;
    float xc       = cam->xc;
    float yc       = cam->yc;
    float c        = cam->c;
    float d        = cam->d;
    float e        = cam->e;
    uint8_t length_pol = cam->length_pol;

    float invdet   = 1/(c-d*e);

    // back-projection of u and v
    *xp = invdet*(    (u - xc) - d*(v - yc) );
    *yp = invdet*( -e*(u - xc) + c*(v - yc) );

    float r = sqrt(SQR((*xp)) + SQR((*yp)));
    *zp       = pol[0];

    float r_i = 1;

    // compute z from polynomial model
    for (uint8_t i = 1; i < length_pol; i++)
    {
      r_i *= r;
      *zp += r_i*pol[i];
    }
}
```

4

2.2 Mapping Optical Flow to Unit Sphere

Now that the model of the omnidirectional camera is known - and the pixels can be mapped onto a plane or a sphere -, we can deduce the viewing direction corresponding to a given pixel. However, the particular geometry of the omnidirectional camera induces high distortions in the image. Hence, the optical flow, which is computed using Lucas-Kanade method, has to be projected onto the unit sphere using the method presented in [6]. Mapping the optical flow directly on the unit sphere allows us to simplify further processing required for the estimation of the location of the focus of expansion, through the straightforward use of spherical coordinates. However, though more natural, the sphere still induces radial distortions and may not be appropriate for all types of motions, as exposed by Shakernia et al.

The optical rays are described in spherical coordinates (ρ, θ, Φ) from the center of the camera, where ρ is the magnitude, θ is the azimuth in the X-Y plane, and Φ is the polar angle between the ray and the Z-axis. Given an image point $(u, v)^T$, we first compute the corresponding ray (or back-projection ray) $b = (x, y, z)^T$ using equations (1) and (2), and normalize it to unit length $s = b/\|b\|$. The spherical coordinates of the "unitized" back-projection ray are given by:

$$\begin{cases} \rho = 1 \\ \theta = \arctan(\dfrac{y}{x}) \\ \Phi = \arctan(\dfrac{r}{z}) \end{cases} \tag{3}$$

where $r = \sqrt{x^2 + y^2}$. Using equations (1) and (2), we can derive the following Jacobian which relates the partial derivatives from the image plane to the unit sphere:

$$J = \begin{bmatrix} \dfrac{\partial \theta}{\partial u} & \dfrac{\partial \theta}{\partial v} \\ \dfrac{\partial \Phi}{\partial u} & \dfrac{\partial \Phi}{\partial v} \end{bmatrix} = \frac{1}{det(A)} \begin{bmatrix} \dfrac{-y - ex}{r^2} & \dfrac{dy + cx}{r^2} \\ \dfrac{d\Phi}{dr} \dfrac{(x - ey)}{r} & \dfrac{d\Phi}{dr} \dfrac{(cy - dc)}{r} \end{bmatrix} \tag{4}$$

where $A = \begin{bmatrix} c & d \\ e & 1 \end{bmatrix}$ and $\dfrac{d\Phi}{dr} = \dfrac{z - rf'(r)}{r^2 + z^2}$

Then, we compute the transformation which takes the partial derivatives on the unit sphere from spherical coordinates to rectangular coordinates:

$$S = \begin{bmatrix} -\sin\theta\sin\Phi & \cos\theta\cos\Phi \\ \cos\theta\sin\Phi & \sin\theta\cos\Phi \\ 0 & -\sin\Phi \end{bmatrix} \tag{5}$$

Therefore, we can compute the mapping of an image point $(u, v)^T$ (considered distorted) and its corresponding optical flow $(\dot{u}, \dot{v})^T$ to the unit sphere:

$$s = b/\|b\| \qquad\qquad \dot{s} = SJ \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \tag{6}$$

A possible C implementation is detailed in Code 2

Code 2: Backprojection of optical flow

```c
void flow2world(float *flow_x, float *flow_y, float *flow_z,
    float xp, float yp, float zp, float flow_u,
    float flow_v, cam_model *cam)
{
    float *pol     = cam->pol;
    float c        = cam->c;
    float d        = cam->d;
    float e        = cam->e;
    uint8_t length_pol = cam->length_pol;
    float invdet  = 1/(c-d*e);

    // from cartesian to spherical coordinates
    float r     = sqrt(SQR(xp) + SQR(yp));
    float theta = atan2(yp,xp);
    float phi = atan2(r, zp);

    // compute polynomial model derivative
    float r_i = 1;
    float dzp = pol[1];
    for (uint8_t i=2; i < length_pol; i++)
    {
      r_i *=r;
      dzp += i*pol[i]*r_i;
    }
    // project optic-flow on unit sphere
    float d_thetau = invdet*(-yp/SQR(r) - e*xp/SQR(r));
    float d_thetav = invdet*(d*yp/SQR(r) + c*xp/SQR(r));
    float d_phir = (zp - r*dzp)/(SQR(zp) + SQR(r));
    float d_phiu = d_phir*invdet*(xp/r - e*yp/r);
    float d_phiv = d_phir*invdet*(-d*xp/r + c*yp/r);
    float flow_theta = d_thetau*flow_u
            + d_thetav*flow_v;
    float flow_phi = d_phiu*flow_u + d_phiv*flow_v;

    // from spherical to cartesian coordinates
    *flow_x = -sin(theta)*qsin(phi)*flow_theta
        + cos(theta)*cos(phi)*flow_phi;
    *flow_y = cos(theta)*sin(phi)*flow_theta
        + sin(theta)*cos(phi)*flow_phi;
    *flow_z = -sin(phi)*flow_phi;
}
```

2.3   Derotation of Optical Flow

Under the assumption of differential motion, that is small translation and rotation of the camera between each frame, we can approximate the geometrical effects of egomotion on the perceived image motion to a first-order Taylor polynomial. Hence, as the scene is projected on the unit sphere centered on the vantage point, each optical flow measurement can be expressed as a 3D vector tangent to the unit sphere and perpendicular to the viewing direction:

$$p = -w \times d - \frac{v - (v \cdot d)d}{D} \tag{7}$$

where d is a unit vector describing the viewing direction, w the angular speed vector, v the translational velocity vector and D the distance to the viewed object. The measured optical flow p can be decoupled and expressed in two parts, namely the rotation-induced and the translation-induced optical flows, respectively p_r and p_t.

As we are only interested in the estimation of the angle of attack, i.e. the direction of translation, only the translational part p_t of the optical flow is required. Hence, we proceed to the removal of the rotational part p_r of the optical flow through the process of derotation [4]. If necessary, the main axis Z of the camera can be automatically calibrated using the method proposed in [2]. Even after calibration, the derotation procedure may introduce additional noise to the optical flow measurement, all the more so that the rotational component is larger than the translational part of the optical flow. This may be exacerbated by the sensor bias compensation and noise reduction only achieved using averaging and low-pass filtering.

However, before proceeding to the derotation of the optical flow, we must first consider scaling the angular rate values. Indeed the optical flow is given in pixels/frame or counts/frame whereas the gyroscopes measurements are rad/s. Hence, we need to estimate the scaling factor between both measurements. This can be achieved through regression from measurements obtained for pure rotations of the drone. To simplify both the procedure and the processing of the measurements, we only sampled the optical flow measurements at the center of the camera image. This allowed us to directly use the raw measurements from the optic-flow sensor and the gyroscopes. The measurements were first verified so as to remove obvious outliers and the regression is perfomed by computing the mean, standard deviation (SD) and standard error (SE) of the ratio flow/gyro. Then, we also apply least square regression (both with and without bias) on the same dataset. Note that, prior to regression, the gyroscope measurements were first multiplied by 0.01 so as to obtain comparable values (in terms of magnitude). The results from regression are displayed in Table 1.

The results show a high standard deviation due to high noise in the optic-flow measurements. The low resolution and the sometimes uniform textures of the environment in which the data were sampled can be accounted for the high uncertainty in the scaling factor. Interestingly enough, this significantly impact the results of the full least square regression. Finally, the results from least square regression without bias conforts us in the magnitude of the scaling factor with only 2% average error on its value. Therefore, as we are facing very noisy data, we kept the scaling factor to 0.01 for the remaining of

8

Method	Angular rate scaling factor ($\times 0.01$)		
Mean Ratio (flow/gyro)	Mean	SD	SE
	1.07	0.427	0.0288
Least Squares (full)	Bias	Weight	RSS
	0.00192	0.826	0.00152
Least Squares (no bias)	Weight		RSS
	0.984		0.00163

Table 1: **Results from calibration of the angular rate scaling factor**-Based on 220 samples taken at the center of the camera image

the project.

2.4   Finding the Focus of Expansion

In order to find the direction of translation, we use the derotated optical flow to constrain the translation direction to lie on a great circle. This great circle is the intersection of a plane - defined by the vector describing a viewing direction d, its corresponding optical flow vector p_t and the center of the sphere O - with the unit sphere [1]. Hence, each optic-flow vector gives us a constrain on the position of the focus of expansion (FOE), which can be located anywhere on the great circle. Although only two great circles are required to constrain the location of the FOE to a single point, several measurements are used so as to recover from possible outliers and noise.

For a robust estimation of the translation direction, two different approaches are possible. The first one relies on RANSAC (RANdom SAmple Consensus) and its variants. However, this method implies several estimations of the location of the FOE based on randomly sampled measurements and may not be suitable for implementation on a microcontroller and fast refresh rate. Another approach is to find the best intersection of all the great circle arising from the optical flow measurements such as the Hough-reminiscent voting method proposed in [3]. Voting has the advantage of performing in constant time under increasing outlier proportions without any loss of accuracy. Its higher robustness and the fact that it does not require an optimization process make this method the most suitable for embedded robotics.

In order to simplify the mathematical representation of the great circles, we rely on their normal vectors. Indeed, since a great circle is the intersection of the sphere with a plane passing though its origin, it can be defined using one of the unit normal vectors. Formally, for a unit vector $n = (n_x, n_y, n_z)^T$, a great circle on the unit sphere S^2 is defined as

$$C = S^2 \bigcup \{x | x^T n = 0, x \in \mathbb{R}^3\} \tag{8}$$

---

[1]Indeed, the plane contains the viewing direction vector and the optic-flow sensor

This normal vector n is readily computed using the cross-product,

$$n = \frac{d \times p_t}{\|d \times p_t\|} \tag{9}$$

Thus, we can vote along a great circle, given its normal vector, by computing simple dot products (i.e. 3 multiplications and 2 additions).

In practice, the intersection of two great circles define 2 possible direction of translations. To disambiguate between the two, one may pick one of the two points and calculate the angle between it and the vector p_t. If the angle is larger than $\pi/2$ radians, then that point is the focus of expansion. Another possibility is to use an additional sensor (such as a sonar) to choose the correct direction. In order to improve the processing speed on the camera chip (and to avoid ambiguity), we only consider one hemisphere, that is half the number of possible directions of translation. In other words, we only determine the axis of translation rather than its direction and we work on the hemisphere defined by $z < 0$.

The voting procedure requires the storage of evenly distributed vectors on the unit sphere. The minimal angle between these vectors define the resolution of the voting procedure. For tractability reasons, we cannot afford to vote on the complete unit sphere in one batch. Hence, we are considering an iterative method which can be divided in two phases. The first one is a coarse voting procedure where we compute a rough estimate of the direction of translation using a few directions sampled from one hemisphere. This estimate is then used in the next phase where we perform voting on a refined spherical grid of higher resolution but reduced spread. The complete voting procedure is detailed in Algorithm 1.

Algorithm 1: Hough-reminiscent voting

1 Select optic flow measurement p_t and viewing direction d
2 Compute normal vector $n = \dfrac{d \times p_t}{\|d \times p_t\|}$
3 for i = 1 to N_{coarse} do
4   if $x_i^T n = 0$ then
5   Vote for direction x_i
6   end if
7 end for
8 Find the bins with maximum votes. This gives the coarse estimate.
9 Rotate the refined bins to center them on coarse estimate
10 for j = 1 to $N_{refined}$ do
11   if $x_j^T n = 0$ then
12   Vote for direction x_j
13   end if
14 end for
15 Find the bins with maximum votes. This gives the fine estimate

---

The refined voting requires rotating the refined bins, which we computed for the a

predefined direction (in our case $z = (0, \ 0, -1)$). There are three ways to achieve this transformation, namely the rotation matrix, axis-angle and quaternion methods. From a computational point of view, the rotation matrix method is the cheapest one (6 additions and 9 multiplications) whereas the quaternion method is the most expensive (18 additions and 21 multiplications). However, it is easier for us to use the axis-angle representation and than convert it to a rotation matrix so as to perform the transformation on every bins. The latter convertion accounts for 13 additions and 15 multiplications but the subsequent processing will only require 6 additions and 9 multiplications, provided that we can store the entire rotation matrix. A C implementation is given in Code 3.

Obviously, this whole method is sensible to noise which could affect the voting procedure if the accumulator has a finite resolution. In our case, the coarse resolution was set to 31° and the fine resolution was chosen as approximately 4°. The latter value is justified considering the resolution of the camera (which is around 1.5/pixel) and the additional noise induced by derotation. The former value was chosen so as to limit the number of coarse bins. The voting bins were evenly distributed on the unit and generated using the icosahedron method. Table 2 summarizes the voting bins parameters.

Voting bins parameter			
Voting set	Region	Number of bins	Resolution (°)
Coarse	Hemisphere	25	31
Refined	Coarse estimate	42	4

Table 2: Voting bins parameters-The voting bins are evely distributed on the unit sphere and were generated using icosahedron method.

Code 3: Hough-reminiscent voting

```c
void rotate_bins(voting_bins *bins, float best_x,
                 float best_y, float best_z,
                 float *r_dir_x, float *r_dir_y,
                 float *r_dir_z)
{
    if(best_x==0.0f, best_y==0.0f, best_z==-1.0f){
        for(uint8_t i=0; i<bins->size; i++){
            bins->x[i] = r_dir_x[i];
            bins->y[i] = r_dir_y[i];
            bins->z[i] = r_dir_z[i];
        }
    }
    else{
        // store cosine and sine
        float c = -best_z;
        float s = maths_fast_sqrt(1 - SQR(c));

        //store axis
        float u_x = 1.0f*best_y;
        float u_y = -1.0f * best_x;
        float u_z = 0.0f;

        // compute rotation matrix
        float R[9];
        aa2mat(R, u_x, u_y, u_z, c, s);

        // compute rotated refined bins
        for(uint8_t i=0; i<bins->size; i++){
            bins->x[i] = r_dir_x[i]*R[0]+r_dir_y[i]*R[1]
                        +r_dir_z[i]*R[2];
            bins->y[i] = r_dir_x[i]*R[3]+r_dir_y[i]*R[4]
                        +r_dir_z[i]*R[5];
            bins->z[i] = r_dir_x[i]*R[6]+r_dir_y[i]*R[7]
                        +r_dir_z[i]*R[8];
        }
    }
}
```

2.5 Egomotion from Optical Flow Measurements

Another solution to our problem is to estimate the direction and the speed (scale) of translation. This can be achieved using EKF-based methods such as the one proposed in [1], which allows for more parameters to be estimated through the fusion of sensor measurements from the Inertial Measurement Unit (IMU) and the optic-flow sensor.

Considering the available hardware, this method would require the transmission of the preprocessed data from the camera chip to the autopilot. Hence, it would be necessary to estimate the additional delay due to preprocessing and transmission. In this case, the computation of optical flow measurements and viewing direction, the projection on the unit sphere, and the derotation, could be performed on the camera chip while the remaining computations (EKF predictions and updates) would be realized by the autopilot.

Unfortunately, this method is still sensitive to outliers and would benefit from the addition of a random sampling step (RANSAC) and/or a quality factor to remove erroneous optical flow measurements.

References

[1] A. Briod, J.-C. Zufferey, and D. Floreano, "A method for ego-motion estimation in micro-hovering platforms flying in very cluttered environments," *Autonomous Robots*, 2015.

[2] A. Briod, J.-C. Zufferey, and D. Floreano, "Automatically calibrating the viewing direction of optic-flow sensors," in *Robotics and Automation (ICRA)*, pp. 3956–3961, 2012.

[3] J. J. K. Lim, *Egomotion Estimation with Large Field-of-View Vision*. PhD thesis, 2010.

[4] M. V. Srinivasan, S. Thurrowgood, and D. Soccol, "From Visual Guidance in Flying Insects to Autonomous Aerial Vehicles," in *Flying Insects and Robots*, pp. 15–28, 2009.

[5] M. A. Scaramuzza, D. and R. Siegwart, "A flexible technique for accurate omni-directional camera calibration and structure from motion," in *Proceedings of IEEE International Conference of Vision Systems*, 2006.

[6] O. Shakernia, R. Vidal, and S. Sastry, "Omnidirectional egomotion estimation from back-projection flow," in *Computer Vision and Pattern Recognition Workshop*, pp. 82–82, IEEE, 2003.