Methods for Movement Trajectory Determination in Space

Eriks Klavins
Riga Technical University

***Abstract* – Unmanned aerial vehicles have become more widely used for military, security, building inspection etc. Flight path and collision avoidance are the main research tasks for UAV control. This paper serves as background for a new method, which can be used for UAV onboard trajectory determination. Collecting UAV control methods and their advantages is possible to create one method that collects advantages of various control methods. Trajectory is established by using the image sensor and image processing methods. New method can be used to cancel out inertial navigation system errors like GPS does.**

***Keywords* – Flight trajectory, inertial navigation system, optical flow, unmanned aerial vehicle.**

I. Introduction

Unmanned aerial vehicles are used almost everywhere. In recent years, unmanned aerial vehicle (UAV) has become a major focus of active research, since it can extend our capability in a variety of areas, especially for applications, such as search-and-rescue and inspection. UAVs tend to fly in open sky, far from any obstacles and rely on external beacons – mainly GPS – to localize them-selves and navigate.

The control of quadcopter during autonomous flights relies on knowledge of variables such as position, velocity and orientation, which can be partly calculated using information provided by on-board inertial sensors. Knowledge of environment can also be gained from visual sensors, such as camera. Using optical flow and other image processing methods, it is possible to detect and recognize obstacles and avoid collisions with them.

Another task of UAVs is to cooperate to solve some tasks especially at low altitude where GPS cannot be used for such flights. We can take inspiration from flies and bees, study their sensor suites and ways of processing information in order to extract principles that could then be applied to UAV. It turned out that insects are mainly relying on low-resolution, monocular vision [1], inertial and airflow sensors [2] to control their flight. This is interesting because the corresponding sensors are now commercially available with small, light packaging, and extremely low power. Therefore, rather than opting for bulky active 3D range finders weighing a few kilograms, dynamic flight in the vicinity of obstacles can be achieved with far lower weight by using passive sensors such as vision, microelectromechanical system (MEMS) rate gyros and miniature anemometers [3].

UAV movement trajectory detection is one of main tasks for localization, collision avoidance and other research topics. Flight trajectory can be calculated by various methods. In this paper, some methods are presented to find a flight trajectory. All methods are compared, and their advantages and disadvantages are also determined. According to advantages of the methods studied, a new method is proposed for flight trajectory determination. In this section, we describe the main problem to be solved: flight trajectory determination. According to advantages of all methods, an embedded solution to onboard flight trajectory detection is introduced with the aim to be used for indoor and outdoor flights. In obstacle avoidance and cooperative flight tasks is used flight trajectory of UAV to find new flight trajectory avoiding collisions with other UAVs and with obstacles.

II. Existing Methods

A. Inertial Navigation System

As mentioned before, all unmanned aerial vehicles contain Inertial Measurement Unit (IMU). IMU typically contain three orthogonal rate-gyroscopes and three orthogonal accelerometers, measuring angular velocity and linear acceleration respectively. Some IMUs include three orthogonal magnetic field sensors to find the yaw angle according to North. By processing signals from these sensors, it is possible to track the position and orientation of a device. Inertial navigation is used in a wide range of applications, including the navigation of aircraft, tactical and strategic missiles, spacecraft, submarines and ships. To track the position of an INS, the acceleration signal $a_b(t)$ obtained from the accelerometer is projected into the global frame of reference:

$$a_b(t) = \begin{bmatrix} a_{bx}(t) \\ a_{by}(t) \\ a_{bz}(t) \end{bmatrix} \quad (1)$$

$$a_g(t) = C(t)a_b(t) \quad (2)$$

where $C(t)$ is UAV attitude at time t. Acceleration due to gravity is then subtracted and the remaining acceleration is integrated once to obtain velocity, and again to obtain displacement:

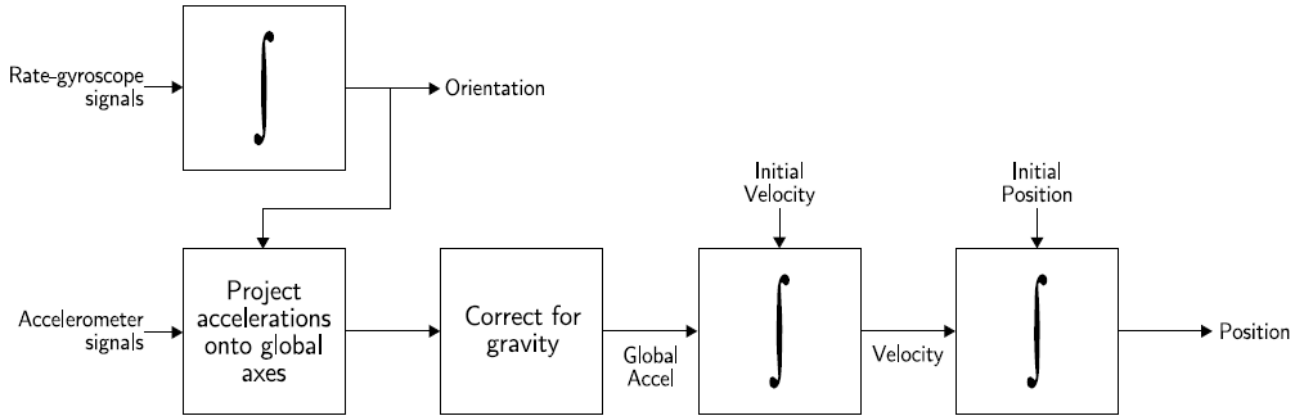

Fig. 1. Strap-down inertial navigation algorithm.

$$a_g(t) = V_g(0) + \int_0^t a_g(t) - g_g dt, \quad (3)$$

$$s_g(t) = s_g(0) + \int_0^t V_g(t) dt \quad (4)$$

where $V_g(0)$ is the UAV initial velocity, $s_g(0)$ is the UAV initial displacement and g_g is the acceleration due to gravity in the global frame [4].

For MEMS devices, angle random walk (noise) and uncorrected bias errors are typically the error sources which limit the performance of the device; however, the relative importance of each error source depends on the specific sensor being used. Fig. 1 displays an INS algorithm. As it can be seen, errors in the accelerations and angular rates lead to steadily growing errors in position and velocity components of the aircraft, due to integration. These are called navigation errors and there are nine of them – three position errors, three velocity errors, two attitude errors and one heading error. If an unaided INS is used, these errors grow with time. It is for this reason that the INS is usually aided with either GPS, Doppler heading sensor or air-data dead reckoning systems as it can be seen in Fig. 2. INS uses attitude and heading reference system output data to calculate speed, path and other outputs. As it can be seen in Fig. 2, GPS outputs are used in INS to correct calculation errors.

B. Camera Sensor Network

Consisting of a large number of low-power camera nodes, visual sensor networks support a great number of novel vision-based applications. The camera nodes provide information from a monitored site, performing distributed and collaborative processing of their collected data. Using multiple cameras in the network provides different views of the scene, which enhances the reliability of the captured events. However, the large amount of image data produced by the cameras combined with the network resource constraints requires exploring a new means for data processing, communication, and sensor management. Localization is a well-established problem in sensor networks.

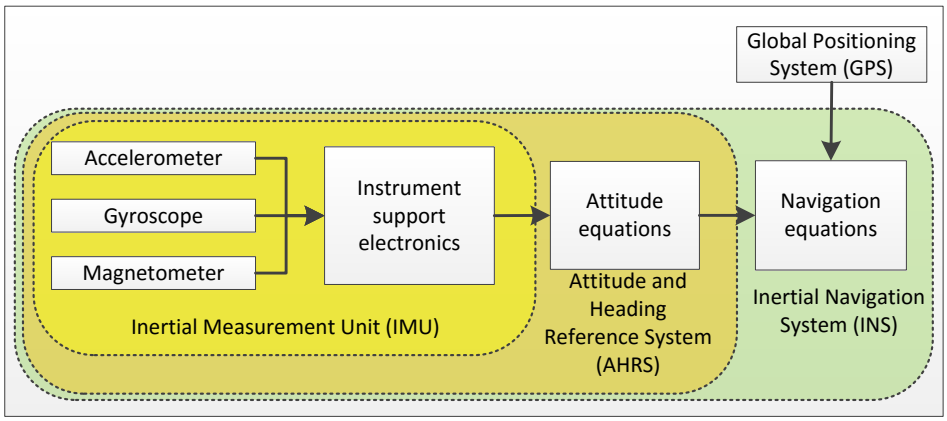

Fig. 2. Inertial Navigation System error corrected by GPS.

Algorithms for localization can be categorized into two main classes, namely, fine-grained algorithms and coarse-grained algorithms. *Fine-grained* methods use timing and/or signal strength for localization. In this class of localization methods, only a few sensor positions are known. These sensors are called *beacons* or *anchors*. The knowledge of beacons is then propagated across the entire network to find the position of the remaining sensors [5].

Coarse-grained algorithms can further be divided into *non-statistical* and *statistical* approaches. Non-statistical approach is used to calibrate a network of randomly placed cameras with non-overlapping fields of view using moving scene features in the near and far fields [6]. Statistical approaches include numerical, motion-based, maximum a posteriori (MAP), and velocity extrapolation based approaches. Numerical solutions are iterative methods for network localization, and each iteration contributes to the reduction of the residual errors. Existing approaches are generally variations of the gradient descent or the Gauss-Newton methods [5], [7].

In structure from motion (SFM), the trajectory of a moving camera, and the 3D coordinates of a stationary target are recovered simultaneously from a series of 2D images of a scene. In [8], the focus is on real-time processing of the image data using an extended Kalman filter. Similar to SFM, simultaneous localization and mapping (SLAM) localizes a moving sensor (robot) and estimates its trajectory using its ego-motion and the stationary objects in the scene. [6]

Let us suppose we have a network of N non-overlapping cameras $\psi = \{C_1, C_2, \dots, C_n\}$, similar to [6]. Let a trajectory Υ_i within C_i be represented as $\Upsilon_i = (x_i, y_i)$ where (x_i, y_i) is the estimated position of the target in the image plane from camera C_i. Furthermore, let each observation be generated by a motion model as

$$\begin{bmatrix} x_i \\ x_i^v \\ y_i \\ y_i^v \end{bmatrix}' = \begin{bmatrix} 1 & a_x & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & a_y \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ x_i^v \\ y_i \\ y_i^v \end{bmatrix} + \begin{bmatrix} v_x \\ v_x^v \\ v_y \\ v_y^v \end{bmatrix}, \quad (5)$$

where (x_i^v, y_i^v) is the velocity of the object on x and y axis. In addition, (a_x, a_y) may change over a period of time with covariance. Moreover, $\mathrm{v}(\mathcal{N}(0, \Sigma_v))$ is modeling additive noise with covariance $\Sigma_v = I^{4\times4} * 1e-3$, where I is identity matrix. If each camera C_i provides a vertical top-down view of the scene (i.e., its optical axis is perpendicular to the ground), the number of parameters for the localization of each camera C_i is reduced to two, namely, the camera C_i position

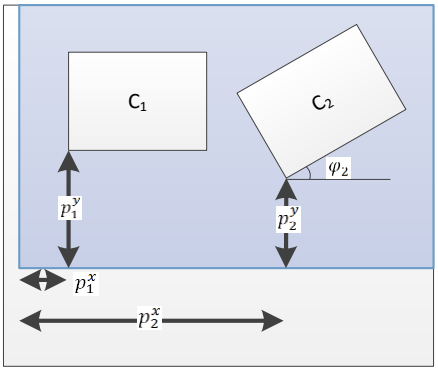

Fig. 3. Camera view plane by coordinates and angles according to the whole environment.

$P_i = (p_i^x, p_i^y)$ and the rotation angle, φ_i expressed as the relative angle between the camera C_i and the horizontal axis (see Fig. 3). To summarize, the parameters Ξ_i for camera C_i are

$$\Xi_i = [p_i^x \quad p_i^y \quad \varphi_i] \quad (6)$$

If C_i observes the object at a particular time instant t and after $t + \tau$ time intervals the object enters into $C_{i+\eta}$ with $C_i \neq C_{i+\eta}$ then it can be visualized as $C_{i+\eta}$ views the object from the $\varphi_{i,i+\eta}\Upsilon_{i,i+\eta}$ position, where $\varphi_{i,i+\eta}$ and $\Upsilon_{i,i+\eta}$ are the rotation matrix and the translation vector, respectively.

The camera *localization* process estimates $(\varphi_{i,i+\eta}, \Upsilon_{i,i+\eta})$ so that the configuration estimation error ε reaches minimum, that is

$$\left(\varphi_{i,i+\eta} + \Upsilon_{i,i+\eta}\right) \begin{bmatrix} x_{i+\eta}(t+\tau) \\ y_{i+\eta}(t+\tau) \end{bmatrix} - \begin{bmatrix} \hat{x}_{i,i+\eta}(t+\tau) \\ \hat{y}_{i,i+\eta}(t+\tau) \end{bmatrix} = \varepsilon \xrightarrow{yields} 0, \quad (7)$$

where $\hat{x}_{i,i+\eta}(t+\tau), \hat{y}_{i,i+\eta}(t+\tau)$ is the projected estimate of the object position from C_i at t to $C_{i+\eta}$ at $t + \tau$.
If all cameras are located randomly, then according to equation (6) each camera C_i location is calculated by (9) with parameters of (8) [5].

$$\Xi_i = [p_i^x \quad p_i^y \quad p_i^z \quad \varphi_i \quad \theta_i \quad \omega_i] \quad (8)$$

$$\begin{bmatrix} \varphi_{i,i+\eta} + \Upsilon_{i,i+\eta} \\ \theta_{i,i+\eta} + \Upsilon_{i,i+\eta} \\ \omega_{i,i+\eta} + \Upsilon_{i,i+\eta} \end{bmatrix}^T \begin{bmatrix} x_{i+\eta}(t+\tau) \\ y_{i+\eta}(t+\tau) \\ z_{i+\eta}(t+\tau) \end{bmatrix} - \begin{bmatrix} \hat{x}_{i,i+\eta}(t+\tau) \\ \hat{y}_{i,i+\eta}(t+\tau) \\ \hat{z}_{i,i+\eta}(t+\tau) \end{bmatrix} = \varepsilon \xrightarrow{yields} 0 \quad (9)$$

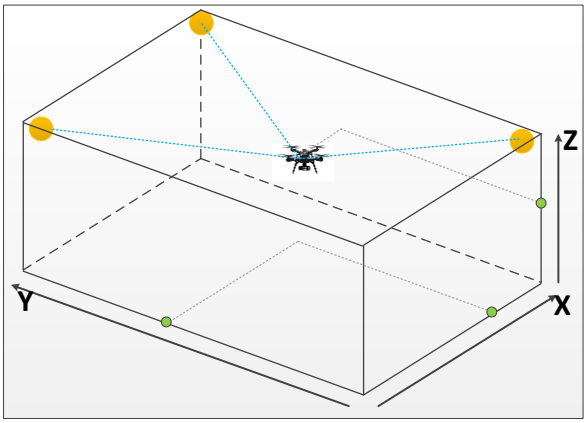

Fig. 4. Room with three cameras (orange dots) locating UAV. Green points show actual UAV position according to room X, Y and Z axis.

III. Optical Flow

UAV control systems become more powerful and can be used for image processing. By processing images, it is possible to detect obstacles, avoid them, localize and build a 3D environment map. UAV is used to fly in various environments. As mentioned before, insects use vision to avoid collisions with obstacles. Optical flow is one of principles, how image processing can help us to determine flight direction. There are various methods for optic flow calculation.

A. The Lucas-Kanade Method

The Lucas-Kanade method assumes that the displacement of the image contents between two nearby frames is small and approximately constant within a neighborhood of the point O under consideration. The image flow vector (V_x, V_y) must satisfy

$$I_x(q_i)V_x + I_y(q_i)V_y = -I_t(q_i), \tag{10}$$

where q_i is pixel, $I_{x,y,t}$ are pixel intensity at x, y coordinates and time t [9]. For multiple windows, equation (10) can be written as

$$\begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_i) & I_y(q_i) \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} -I_x(q_1) \\ -I_x(q_2) \\ \vdots \\ -I_t(q_i) \end{bmatrix} \tag{11}$$

or as $AV = B$. Optical flow vector can be calculated by

$$\begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i)I_y(q_i) \\ \sum_i I_x(q_i)I_y(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} * \\ * \begin{bmatrix} -\sum_i I_x(q_i)I_t(q_i) \\ -\sum_i I_y(q_i)I_t(q_i) \end{bmatrix} = V \tag{12}$$

B. The Horn-Schunck Method

The Horn-Schunck method assumes smoothness in the flow over the whole image. Thus, it tries to minimize distortions in flow and prefers solutions which show more smoothness. The flow is formulated as a global energy functional which is then sought to be minimized [9]. This function is given for two-dimensional image streams as

$$E = \iint \left[\left(I_x u + I_y v + I_t \right)^2 \right] + \\ + \alpha^2 (\|\nabla u\|^2 + \|\nabla v\|^2) dx dy \tag{13}$$

$$V = [u(x,y), v(x,y)]^T \tag{14}$$

where α is regulation constant. This method depends on the neighboring values of the flow field; it must be repeated once the neighbors have been updated. The following iterative scheme is derived

$$u_{k+1} = u_k - \frac{I_x(I_x u_k + I_y v_k + I_t)}{\alpha^2 + I_x^2 + I_y^2} \\ v_{k+1} = v_k - \frac{I_y(I_x u_k + I_y v_k + I_t)}{\alpha^2 + I_x^2 + I_y^2} \tag{15}$$

INS and camera sensor network are calculating UAV actual position. It is still possible to determine flight trajectory by taking its previous positions [9] and flight trajectory can be expressed as

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}' = \begin{pmatrix} x_t + \sigma x \\ y_t + \sigma y \\ z_t + \sigma z \end{pmatrix} + \begin{pmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \end{pmatrix}, \tag{16}$$

where x_t, y_t and z_t are actual UAV positions, $\varepsilon_{x,y,z}$ is error and dx, dy and dz are calculated by

$$\sigma x = \frac{\sum_{k-i}^{k} x_i}{i}, \sigma y = \frac{\sum_{k-i}^{k} y_i}{i}, \\ \sigma z = \frac{\sum_{k-i}^{k} z_i}{i} \tag{17}$$

IV. Results

All these methods have advantages and disadvantages. IMU has an error which increases at the time. There must be an additional system to correct IMU errors, like GPS. GPS is well suited for flights in opened environments, without any obstacles. Our main task is to make UAV flight in various environments, so the GPS would not be a solution to flight indoors. Camera sensor network is good suited for closed rooms. In a similar way, cooperative localization is proposed in [10]. Each base camera can change its position to extend or change an environment. Although there is a solution to this problem, all cameras used in this solution must have an additional platform to change place. But it cannot be used for flights, when an environment changes. Using UAV as a moving platform, the resulting position will include UAV errors

because of position holding. Optical flow methods have a very large amount of data to be processed. To use optical flow methods in the UAV control system, calculation time must be significantly reduced, but still it should be possible to find optical flow vectors.

Cameras tend to be smaller and faster for various image capturing parameters. The same happens with microprocessors. It is possible to perform some image processing on ARM processors. In [11], a method for UAV onboard self-motion estimation was proposed. Images were processed on the onboard fixed computer with Intel Atom dual core processor. Camera was faced down by 45° with respect to the UAV frame. Other study was performed by [12], where using an onboard visual sensor only vertical speed and yaw angle were calculated. It could operate only along paths it traveled during a human-guided training run. Moreover, these paths could be composed only from straight line segments with a limited length [12].

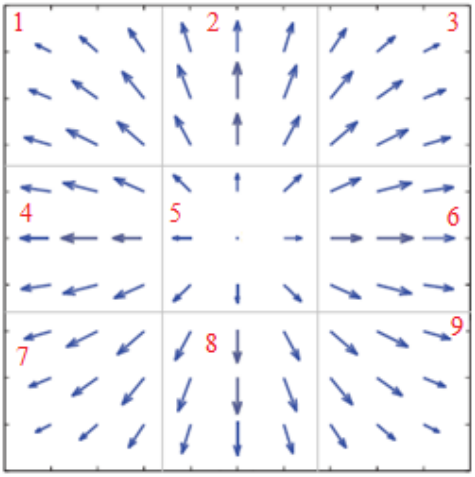

Fig. 5. Optical flow effect vectors divided into nine individual sectors.

In black and white images it is much easier to make image processing. Big resolution creates massive volume of data to be processed. Best image resolution for onboard processing is 640X480 pixels. In such resolution images, there is still enough information about some objects, image depth and etc. The Lucas-Kanade method for optical flow uses multiple scaled images. All these images must be processed to find optical flow. Multiple images use more memory and calculation time.

When UAV flies, main information in pictures is placed at horizontal and vertical axis. It is seen in Fig. 5, where all optical flow vectors are divided into nine groups. Groups with numbers 1, 3, 7 and 9 have additional type of information. These groups in images can be left unprocessed, when it is known that UAV flies straight.

To find a flight trajectory, it is necessary to search for matching pixels in two images, which are taken with some time delay. The second image will have similar information as the first one. Both pictures are processed to find edges using SOBEL operator as it can be seen in Fig. 6 (pictures seen in Fig. 6 are taken by a phone camera). This will significantly reduce pixels with different intensities to two different values: ones and zeroes. Similar image processing was used for object tracking in [13]. Objects were tracked using an optical flow technique.

As mentioned before, these images can be divided into smaller ones. Let us call these picture parts segments. Fig. 5 demonstrates nine different segments with different information about an environment. Central segments contain main changes between both pictures. Fig. 7 displays with yellow lines the rows and columns processed in pictures.

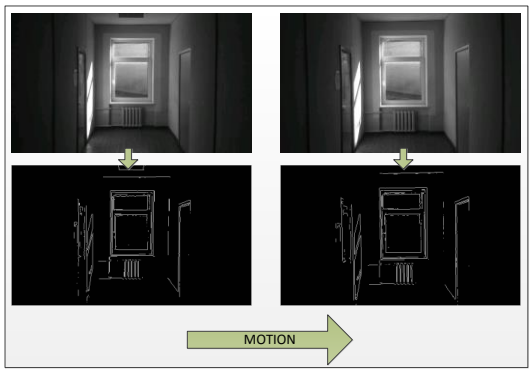

Fig. 6. Two sequential images are processed to find object edges.

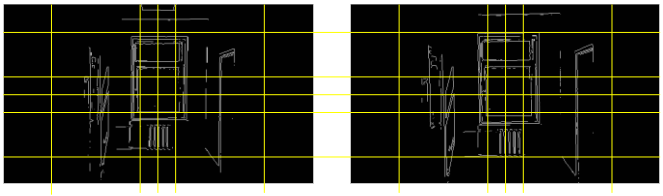

Fig. 7. White pixel search path. Almost all white pixels on yellow lines are registered.

In segments 2, 4, 6 and 8 white pixels are searched three times on rows or columns. Only central segment number 5 is viewed to find white pixels both on rows and columns. Segments 4 and 6 are viewed only by rows. Sectors 1, 3, 7 and 9 are processed as a central segment, but only one time for rows and one time for columns for white pixels.

In some situations after image processing for edge detection, edges may be close to each other. On the next image taken after time, these multiple lines for an edge may be displayed as one line, or vice versa as it is displayed in Fig. 8 by green rectangles. In order to avoid such situations, a minimal difference must be set between white pixels found.

When all white pixel positions are collected in matrices M_{t-1} and M_t for both processed images, these matrices are processed to find differences between white pixels by (18). There must be some method to ensure that white pixels in M_{t-1} and M_t are the same. For an indoor environment it is done easy by straight lines basically of windows and doors. When search is processed, all white pixel coordinates are compared to find a straight line. Difference between those pixel coordinates cannot be bigger than threshold l. Difference between pixels represents an optical flow vector for each sector.

$$A = M_{t-1} - M_t \tag{18}$$

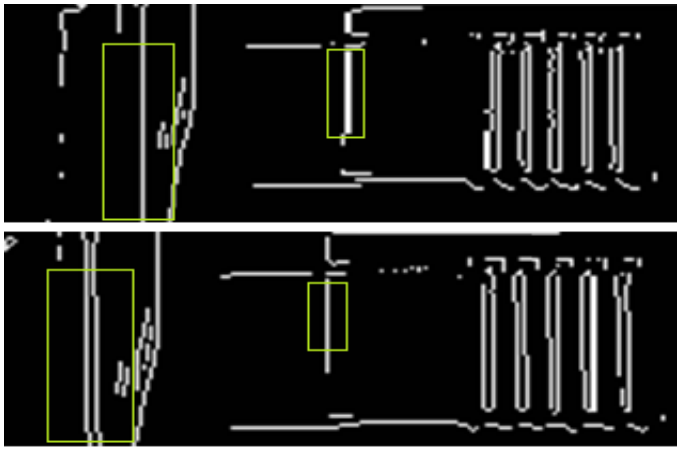

Fig. 8. One and the same edge line on two different images after edge detection.

The flight trajectory will be the sum of vectors from all sectors. By each flight vector calculation, it is possible to replace GPS error correction for INS. For flight at high altitude, this method will not work, except if a camera is faced to the ground.

V. Conclusion

During the present study, three methods have been analyzed, which can be used to determine UAV flight trajectory. All these methods are good for specific solutions. We have presented a new method for UAV flight trajectory determination. This method in three simple steps can find difference between two successive captured images. In the first step, an image must be processed to find object edges. The second step finds object edges, and the third step calculates the traveled path using object edge differences between successive images. For future research, this method should be tested on ARM microcontroller to determine a processing speed. Using this method with INS, it will be possible to increase accuracy of trajectory and remove INS errors as it is done by GPS.

References

[1] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based MAV navigation in unknown and unstructured environments," *Proc. IEEE Int. Conf. Robot. Autom.*, pp. 21–28, 2010. http://dx.doi.org/10.1109/robot.2010.5509920

[2] N. Michael, J. Fink, and V. Kumar, "Cooperative manipulation and transportation with aerial robots," *Auton. Robots*, pp. 1–14, 2010.

[3] J.-C. Zufferey, A. Beyeler, and D. Floreano, "Optic flow to control small {UAV}s," *Work. Vis. Guid. Syst. small Auton. Aer. Veh. (at {IROS}'2008)*, no. c, pp. 4–6, 2008.

[4] J. W. Oliver, "An introduction to inertial navigation," *Arq. Neuropsiquiatr.*, vol. 67, no. 3B, pp. 961–2, 2009.

[5] N. Anjum, "Camera Localization in Distributed Networks Using Trajectory Estimation," *J. Electr. Comput. Eng.*, vol. 2011, pp. 1–13, 2011. http://dx.doi.org/10.1155/2011/604647

[6] N. Anjum and A. Cavallaro, "Automated Localization of a Camera Network," *IEEE Intell. Syst.*, vol. 27, no. 5, pp. 10–18, 2012. http://dx.doi.org/10.1109/MIS.2010.92

[7] C. Taylor, A. Rahimi, J. Bachrach, H. Shrobe, and A. Grue, "Simultaneous localization, calibration, and tracking in an ad hoc sensor network," *Proc. 5th Int. Conf. Inf. Process. Sens. networks*, p. 33, 2006. http://dx.doi.org/10.1109/ipsn.2006.244053

[8] A. Chiuso, P. Favaro, H. Jin, and S. Soatto, "Structure from motion causally integrated over time," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 523–535, 2002. http://dx.doi.org/10.1109/34.993559

[9] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM Comput. Surv.*, vol. 27, no. 3, pp. 433–466, 1995. http://dx.doi.org/10.1145/212094.212141

[10] J. Suh, S. You, and S. Oh, "A cooperative localization algorithm for mobile sensor networks," *2012 IEEE Int. Conf. Autom. Sci. Eng.*, pp. 1126–1131, 2012. http://dx.doi.org/10.1109/coase.2012.6386387

[11] V. Grabe, H. H. Bulthoff, and P. Robuffo Giordano, "Robust optical-flow based self-motion estimation for a quadrotor UAV," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 2153–2159, 2012. http://dx.doi.org/10.1109/iros.2012.6386234

[12] T. Krajník, M. Nitsche, S. Pedre, L. Přeučil, and M. E. Mejail, "A simple visual navigation system for an UAV," *Int. Multi-Conference Syst. Signals Devices, SSD 2012 - Summ. Proc.*, 2012.

[13] S. R. Taichung, "An edge-based approach to improve optical flow algorithm," *Computer (Long. Beach. Calif).*, pp. 45–51, 2010.

Eriks Klavins received Bachelor degree in Computer Science from Riga Technical University in 2010 and Master degree from Riga Technical University in 2012. He undertakes Doctoral studies at RTU Institute of Computer Control, Automation and Computer Science, Department of Computer Networks and System Technology. At present, he works as an Assistant at Riga Technical University.

The Master Thesis was written about image processing on low power 8-bit microcontrollers. The main research focuses on unmanned aerial vehicle onboard control systems which are not dependent on other systems.

E-mail: klavins.eriks@gmail.com; eriks.klavins@rtu.lv

Ēriks Kļaviņš. Kustību trajektorijas noteikšanas metodes

Pētījuma mērķis ir izveidot pamatus jaunai metodei, ar kuras palīdzību būtu iespējams noteikt bezpilota lidaparāta lidojuma trajektoriju. Lai sasniegtu mērķi, tika apskatītas vairākas metodes, ar kuru palīdzību iespējams iegūt līdzīga rakstura informāciju. Metožu pozitīvās pazīmes tiek apkopotas, lai veidotu teorētisku pamatu jaunai metodei. Konkrētā metode balstās uz attēlu apstrādes tehnoloģiju, ar kuras palīdzību tiek noteikts optiskās plūsmas efekts. Šī efekta iegūtie izejas vektori tālāk tiek izmantoti, lai varētu noteikt lidojuma trajektoriju. Metodes ātrdarbības uzlabošanai kameras dati tiek apstrādāti, lai noteiktu objektu robežas. Uz apstrādātiem attēliem tiek noteiktas to atšķirības vairākos attēla segmentos, kuras tiek izteiktas kā vektori. Katrs vektors norāda segmentā izmaiņu virzienu, kurus pēc tam izmanto trajektorijas aprēķināšanai. Precizitātes uzlabošanai jauno metodi būs iespējams izmantot kopā ar inerciālo navigācijas sistēmu, kur tā novērsīs inerciālās navigācijas sistēmas kļūdas.

Ерик Клявиньш. Методы определения траектории движения в пространстве

Целью исследования является установление основ нового метода, с помощью которого можно было бы вычислить траекторию пути полета беспилотного воздушного судна. Для решения задач был исследован ряд способов, с помощью которых можно получить аналогичную информацию. Методы положительных особенностей суммируются для того, чтобы сформировать теоретические основы нового метода. Данный метод основан на технологии обработки изображения, в результате чего можно получить эффект оптического потока. Исходные векторы эффекта используются затем для определения траектории. Чтобы улучшить проведение расчётов, для определения границ объектов обрабатываются данные камеры. В обработанных изображениях определяются различия в ряде сегментов, которые выражаются в виде векторов. Каждый вектор сегмента указывает направления изменений, которые затем используются для расчета траектории. Чтобы повысить точность, новый метод можно будет использовать в сочетании с инерциальной навигационной системой, которая поможет предотвратить ошибки инерциальной навигационной системы.