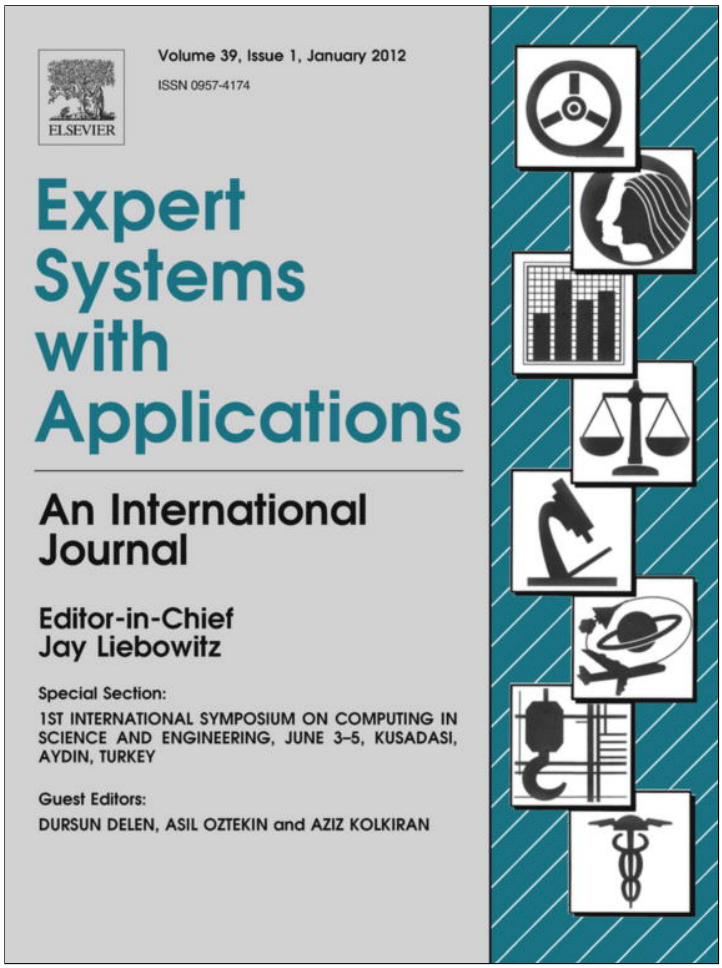

Expert Systems with Applications 39 (2012) 894–905

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Autonomous quadrotor flight with vision-based obstacle avoidance in virtual environment

Aydın Eresen [a], Nevrez İmamoğlu [b], Mehmet Önder Efe [c,*]

[a] Middle East Technical University, Inonu Bulvari, Electrical and Electronics Engineering Department, 06800 Ankara, Turkey
[b] Nanyang Technological University, School of Computer Engineering, 50 Nanyang Avenue, Singapore 639798, Singapore
[c] University of Turkish Aeronautical Association, Department of Pilotage, Akköprü, 06070 Ankara, Turkey

ARTICLE INFO

Keywords:
Optical flow
Vision-based control
Obstacle avoidance

ABSTRACT

In this paper, vision-based autonomous flight with a quadrotor type unmanned aerial vehicle (UAV) is presented. Automatic detection of obstacles and junctions are achieved by the use of optical flow velocities. Variation in the optical flow is used to determine the reference yaw angle. Path to be followed is generated autonomously and the path following process is achieved via a PID controller operating as the low level control scheme. Proposed method is tested in the Google Earth® virtual environment for four different destination points. In each case, autonomous UAV flight is successfully simulated without observing collisions. The results show that the proposed method is a powerful candidate for vision based navigation in an urban environment. Claims are justified with a set of experiments and it is concluded that proper thresholding of the variance of the gradient of optical flow difference have a critical effect on the detectability of roads having different widths.

1. Introduction

Surveillance function is a fundamental capability expected from a UAV and it can be categorized as passive surveillance or active surveillance. Passive surveillance can be achieved by utilizing video frames with the purpose of detecting motion, events or occlusions, segmenting images, recording video data, and the like. Active surveillance systems cover the applications in passive surveillance; further, they respond under certain circumstances, such as giving alarm when fire is detected, producing reference control signals to track objects or to avoid obstacles.

A common application in passive surveillance is to detect and track a moving object. Detection of motion is achieved typically by background modeling and optical flow techniques. Zhang et al. propose a background subtraction technique based on modeling the pixels according to the local dependence histogram and taking the advantage of adaptive threshold algorithm (Zhang, Yao, & Liu, 2008). KaewTraKulPong and Bowden come up with the idea of an improved adaptive background mixture model which detects both motion and shadows (KaewTraKulPong & Bowden, 2001). Spagnolo et al. detect motion using the temporal analysis of the pointwise energy information (Spagnolo, Leo, D'Orazio, Caroppo, & Martiriggiano, 2006). Optical flow is also a useful method in analyzing motion fields on images to detect and segment moving objects. Klappstein et al. achieve moving object segmentation by using the graph-cut algorithm that exploits optical flow information in the analysis of motion (Klappstein, Vaudrey, Rabe, Wedel, & Klette, 2009). Lu et al. use temporal differencing, optical flow, and double background differencing to obtain better results of motion detection with a stationary camera system (Lu, Wang, Yang, & Wu, 2007).

Optical flow computes the motion field through which the changes on the scene can be detected; hence, it enables the detection of a target object from the view of an observer. Detection of an obstacle is the first step of obstacle avoidance in active surveillance systems. Sarcinelli-Filho et al. report a robotic application utilizing the optical flow information to avoid obstacles (Sarcinelli-Filho, Schneebeli, Calderia, & Soria, 2002). Heinrich proposes a fast obstacle detection method combining stereo vision and motion analysis by using both flow and depth information to create a robust detection scheme (Heinrich, 2002). Muratet et al. dwell on a simulation of a collision free autonomous helicopter flight by calculating optical flow and time-to-contact in a virtual environment where a destination point is not defined. Huang et al. demonstrate a vision guided local navigation method that describes a potential field over the robot. Distance and heading to the destination and to the obstacles are used to observe collision free navigation (Huang, Fajen, Fink, & Warren, 2006). Dev et al. introduce a navigation technique for a mobile robot which measures the distance to wall. The orientation of the system and position of the wall are computed by using the spatial derivatives of optical flow (Dev, Kröse, & Groen, 1997). Watanabe et al. use edges of objects to generate a three dimensional obstacle model of the environment, and vision-based

* Corresponding author. Tel.: +90 312 292 4064; fax: +90 312 292 4180.
E-mail addresses: aeresen@gmail.com (A. Eresen), nevrez@gmail.com (N. İmamoğlu), onderefe@etu.edu.tr, onderefe@gmail.com (M. Önder Efe).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.07.087

obstacle avoidance is demonstrated using the developed obstacle model (Watanabe, Johnson, & Calise, 2005). Braillon et al. offer an obstacle detection technique that uses optical flow to define a background model, and as the robot moves, the optical flow matrix is updated according to the measured robot motions. After that, the motion values that are dissimilar to those associated to the ground model are labeled as obstacles (Braillon, Pradalier, Crowley, & Laugier, 2006). Call et al. present an obstacle avoidance method based on the tracking of features. The method detects the corners using Harris corner detector labeling them as features to track, then, features between two consecutive frames are matched and the positions of the obstacles are detected in the subsequent frame (Call, Beard, & Taylor, 2006). Except few ones, considerable number of works reported in the literature focus on straight motion and determining the rotation or speed for obstacle avoidance without having the goal of reaching a specified destination. Duchon compares three methods for maze navigation with only using optical flow in a simulated environment inspiring from insect behavior. Duchon also points out that using the balance theory, or balance strategy, straight motions can be managed at the cost of observing zigzagging in the resulting path (Duchon, 2006) yet the method presented here improves this drawback to some extent.

Autonomous obstacle avoidance systems as in Zhang et al. (2008), KaewTraKulPong and Bowden (2001), Spagnolo, Leo, D'Orazio, Caroppo and Martiriggiano (2006), Klappstein et al. (2009), Lu et al. (2007), Sarcinelli-Filho et al. (2002), Heinrich (2002), Huang et al. (2006), Dev et al. (1997), Watanabe et al. (2005), Braillon et al. (2006), Call et al. (2006), Low and Wyeth (2005), Duchon (2006) require a control system to handle the task after the analysis of the information acquired through vision periphery. Especially, control of UAVs is a challenging research problem for researchers because of the demanding nature of the operating conditions and typically nonlinear and underactuated nature of the vehicle dynamics (Kurnaz, Cetin & Kaynak, 2010). This paper considers the control of rotational and translational behavior of a quadrotor type UAV. The feedback controller is designed to stabilize the dynamics of the quadrotor in Google Earth® virtual environment. According to the outcome of the vision based obstacle detection process, the reference yaw trajectory is derived and it is supplied to the low level control loop. A PID controller is implemented to observe the desired attitude (rotational response) from the vehicle (Ang, Chong, & Li, 2005; Aström & Hagglund, 1995).

The contribution of this paper is to demonstrate that a UAV can perform an autonomous flight toward a predefined destination point by realizing required turns and generating a path using only the vision information. Some studies in the literature use simpler environments with more complicated techniques to reach the specified destination points making the flight costly and unrealistic. The method proposed here utilizes only the optical flow information to detect the junctions and to avoid obstacles. The method proposed needs very small computational time and the overall response of the whole system is suitable for UAV based autonomous navigation applications. This paper is organized as follows: The second section presents the dynamics of the UAV and its physical parameters. The third section presents the optical flow technique and its use in detecting the obstacles in Google Earth® environment. The fourth section is devoted to the implementation specific issues and results. Concluding remarks are given at the end of the paper.

2. Dynamic model of the quadrotor UAV

The quadrotor type UAV, shown in Fig. 1, is a kind of rotating rigid structure having six degrees of freedom. The two pairs of rotors rotate in opposite directions to alleviate the torque imbalance. Therefore first and the third rotors rotate in the counter-clockwise direction while second and fourth ones rotate in the clockwise direction. During the hover condition, varying the angular speeds of each rotor while keeping the changes equal causes a change in the altitude of the vehicle. The motion in the x direction is generated due to change in pitch angle by varying the angular speeds of the first and third rotor inversely proportional. The motion in the y direction is generated due to change in roll angle by varying the angular speeds of the other pair of rotors inversely proportional. Difference in the angular speeds of the two pairs of rotors results in drag torque and yaw motion, which is necessary to perform turns at the junctions. Quadrotor type UAV is modeled under the following assumptions.

(1) The quadrotor structure is rigid and symmetrical.
(2) The center of mass of the quadrotor and body-fixed frame coincides.
(3) Thrust and drag forces are proportional to the square of the angular speeds of the propellers.
(4) The propellers are rigid.
(5) Ground effect is neglected.

Let E and B denote earth fixed frame and body fixed frame respectively as shown in Fig. 1. Let $\{e_{xb}, e_{yb}, e_{zb}\}$ denote the body axes and $\{e_x, e_y, e_z\}$ denote the inertial axes. The dynamic model of the vehicle is derived using Newton–Euler formulation and the equations governing the dynamics can be written in the following form (Amir & Abbass, 2008):

$$\begin{bmatrix} mI_{3\times3} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \dot{\vec{V}} \\ \dot{\vec{\omega}} \end{bmatrix} + \begin{bmatrix} \omega \times m\vec{V} \\ \omega \times I\vec{\omega} \end{bmatrix} = \begin{bmatrix} F \\ \tau \end{bmatrix}, \tag{1}$$

where m is the mass of the vehicle. With the diagonal inertia matrix $I = \mathrm{diag}(I_{xx}\ I_{yy}\ I_{zz}) \in R^{3\times3}$, ω is the body angular velocity, and $\vec{V}$ is the body linear velocity vector. After arranging the terms appropriately, the dynamics in (1) can be cast into six ordinary differential equations (ODEs) as given in (2)–(7). The translational motion is governed by the ODEs in (2)–(4), where it is seen that the Euler angles ϕ (roll), θ (pitch) and ψ (yaw) need to be controlled appropriately to observe a desired motion in Cartesian coordinate system. The latter three ODEs in (5)–(7) govern the rotational dynamics, called the attitude. Observing a desired attitude can be obtained by a properly selected low level controller that provides the control inputs $U2$, $U3$ and $U4$:

$$\ddot{x} = \left(c_\phi s_\theta c_\psi + s_\phi s_\psi\right)\frac{1}{m}U_1, \tag{2}$$

$$\ddot{y} = \left(c_\phi s_\theta s_\psi - s_\phi c_\psi\right)\frac{1}{m}U_1, \tag{3}$$

$$\ddot{z} = -g + \left(c_\phi c_\theta\right)\frac{1}{m}U_1, \tag{4}$$

$$\ddot{\phi} = \dot{\theta}\dot{\psi}\left[\frac{I_{yy} - I_{zz}}{I_{xx}}\right] - \frac{J_r}{I_{xx}}\dot{\theta}\Omega_d + \frac{l}{I_{xx}}U_2, \tag{5}$$

$$\ddot{\theta} = \dot{\phi}\dot{\psi}\left[\frac{I_{zz} - I_{xx}}{I_{yy}}\right] + \frac{J_r}{I_{yy}}\dot{\phi}\Omega_d + \frac{l}{I_{yy}}U_3, \tag{6}$$

$$\ddot{\psi} = \dot{\theta}\dot{\phi}\left[\frac{I_{xx} - I_{yy}}{I_{zz}}\right] + \frac{1}{I_{zz}}U_4, \tag{7}$$

where c_ϕ stands for $\cos(\phi)$ and s_θ denotes $\sin(\theta)$ and the parameters of the vehicle are listed in Table 1. The variable Ω_d seen in roll and pitch dynamics above is defined as in (8). The control inputs U_1, U_2, U_3 and U_4 seen in the ODEs above are defined in (9), where Ω_i is the angular speed (in rad/s) of the ith rotor:

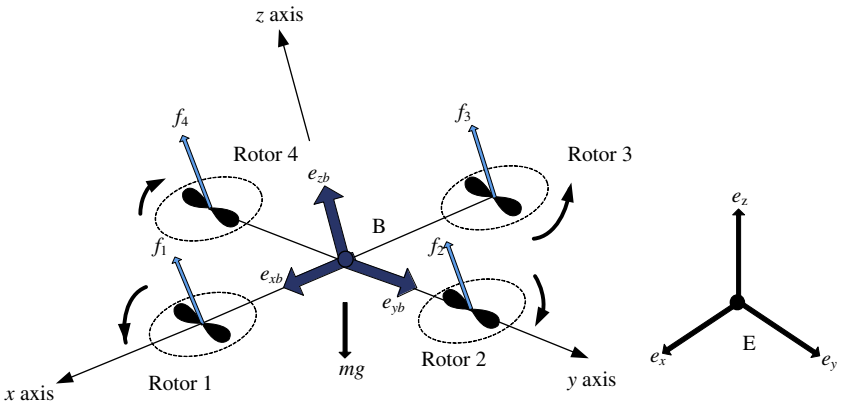

Fig. 1. Structure of quadrotor type UAV.

$$\Omega_d = \Omega_1 - \Omega_2 + \Omega_3 - \Omega_4, \tag{8}$$

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ 0 & -b & 0 & b \\ -b & 0 & b & 0 \\ d & -d & d & -d \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}, \tag{9}$$

where b is the thrust coefficient, d is the drag coefficient. The reader is referred to Castillo, Lozano, and Dzul (2005), Bouabdallah and Siegwart (2007), Chen (2003) for a more detailed discussion on the dynamics of vehicle considered here.

The low level control is implemented by utilizing PID type controllers. Having established the link between Matlab® and Google Earth® environments the dynamics is operated under the Matlab® side and the response is realized in Google Earth® side. While operating this cycle repetitively during the flight, the visual information is acquired and processed at Matlab® side to observe the autonomous flight with obstacle avoidance feature. Such a high level behavioral goal is obtained via generating the necessary control sequences from the PID controllers receiving the command signals produced by the obstacle avoidance scheme (see Fig. 2).

3. Obstacle avoidance

Navigation in a complicated environment is a difficult problem if the flight is desired to be autonomous. Detecting the obstacles such as buildings, holes; pedestrians, vehicles etc. for collision free autonomous flight is the crux of the design problem. In nature, insects such as bees, flies and the like are able to fly using visual sensors, and they are capable of passing through the perceived corridors (Barron & Srinivasan, 2006; Egelhaaf & Kern, 2002; Srinivasan, Zhang, Lehrer, & Collett, 1996). Some studies in biology show that vision systems of flying insects have excessively sensitive to motion as the vision system receives significant amount of excitation from motion thereby providing information about obstacles. After a significant amount of research done in the past, scientists derived a method called *optical flow* to obtain motion flow. Optical flow enables the detection of the movement of brightness in sequentially ordered gray scaled images, e.g., a video stream composed of grayscale images. It also gives information about the motion of the observer as well as the objects in the scene. Optical flow takes the advantage of gradients of the temporal and spatial information to calculate approximate motion vectors, i.e. the temporal derivatives lead to the detection of motion in time domain, and spatial derivatives facilitate the perception of the motion in the two dimensional (2D) coordinate system (Barron, Fleet, & Beauchemin, 1994; Horn & Brian, 1992).

Table 1
Physical parameters of the quadrotor UAV.

Total weight of the vehicle	m	0.800 kg
Gravitational acceleration	g	9.81 kgm^2
Arm length of the vehicle (from c.g. to tip)	l	0.3 m
Moment of inertia along x axis	I_{xx}	15.67×10^{-3} kgm^2
Moment of inertia along y axis	I_{yy}	15.67×10^{-3} kgm^2
Moment of inertia along z axis	I_{zz}	28.34×10^{-3} kgm^2
Thrust factor	b	192.32×10^{-7} Ns^2
Drag factor	d	4.003×10^{-7} Nms^2
Propeller inertia	J_r	6.01×10^{-5} kgm^2

A number of methods to calculate the optical flow are proposed in the literature. The techniques to determine the optical flow can be categorized as differential methods, correlation based methods and frequency based methods (Barron et al., 1994). In frequency based techniques, spatiotemporal velocity-tuned linear filters are utilized to create the new form of the image sequence, and optical flow velocity matrix is obtained from the new form of the image sequence (Fleet & Jepson, 1990). In the correlation based method, features are extracted from sequential images and optical flow is calculated as a matching feature obtained using the consecutive images (Camus, 1997). Differential optical flow techniques take the advantage of spatiotemporal derivatives of image sequences (Barron et al., 1994). Differential-based optical flow algorithms – Lucas–Kanade (Lucas, 1984; Lucas & Kanade, 1981), Horn and Schunk (Barron et al., 1994; Horn & Brian, 1992) – have been experimented in our autonomous flight applications, and due to its prominent features, Horn and Schunk algorithm is chosen to determine the optical flow.

In this paper, autonomous flight is obtained via detecting the junctions, which is necessary to perform collision free turns, and approaching a predefined destination by continuously checking a performance measure. Contrary to the cited body of literature, which focuses on where the obstacles are, the process of flight here is involved with the detection of volumes that are not occupied by obstacles. Detecting the junctions by the UAV is achieved by an algorithm seeking for the least magnitude optical flow field and with a proper treatment of the information contained in the variance of the optical flow velocity the vehicle performs the commanded turns. In the sequel, we summarize the calculation of the optical flow by using the Horn and Schunk method.

3.1. Calculation of optical flow using Horn & Schunk method

In differential optical flow calculation methods, velocity is computed using spatiotemporal derivatives or filtering of the grayscale image. Horn and Schunk improved differential technique using

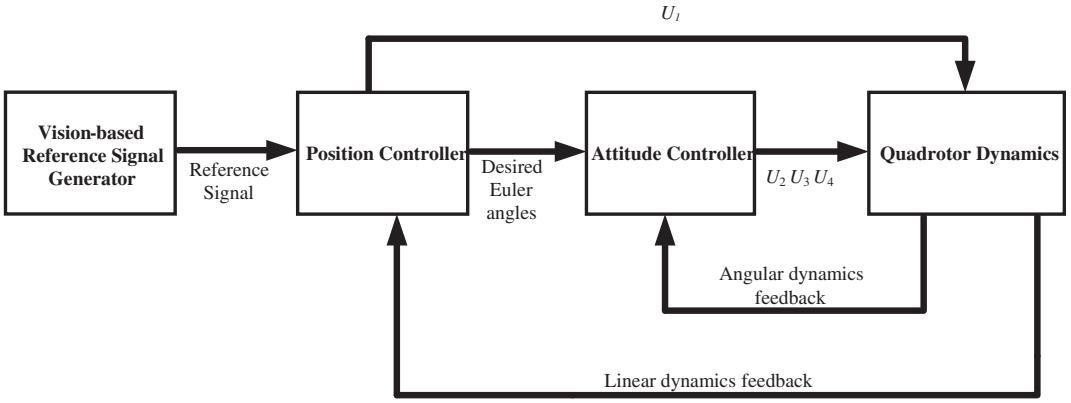

Fig. 2. Quadrotor control block diagram.

additional constraints to obtain more accurate optical flow velocity than differential based approach. It is more sensitive to environmental changes than the method by Lucas and Kanade (Barron et al., 1994), but the sensitivity provides us the necessary information to locate the junctions or obstacles. Hence, in our application, Horn and Schunk method is preferred for computing the optical flow vectors (Horn & Brian, 1992).

Assuming $I(x, y, t)$ as the grayscale density function (intensity image), if the Taylor series expansion of gray scale density function is evaluated, we obtain the expression in (10):

$$I(x+\delta x, y+\delta y, t+\delta t) = I(x,y,t) + I_x\delta x + I_y\delta y + I_t\delta t + \kappa, \tag{10}$$

where I_x and I_y are spatial partial derivatives respectively along x, and y directions and I_t stands for the temporal derivation of intensity image with respect to time, and κ denotes the higher order terms of Taylor series expansion, δx, δy, δt are very small changes in x, y, and t. Higher order terms can be neglected because these terms are small in magnitude, and the equation can be written as in (11), which is indeed an approximation:

$$I(x+\delta x, y+\delta y, t+\delta t) \cong I(x,y,t) + I_x\delta x + I_y\delta y + I_t\delta t. \tag{11}$$

Considering $\delta x \to 0$, $\delta y \to 0$, $\delta t \to 0$ the expression in (11) can be simplified as follows:

$$I_x\frac{dx}{dt} + I_y\frac{dy}{dt} + I_t = 0. \tag{12}$$

Rearranging (12) yields:

$$I_t = -\left(I_x\dot{x} + I_y\dot{y}\right), \tag{13}$$

where $\dot{x}, \dot{y}$are the time derivatives of x, and y. Velocity vector in x and y directions can be expressed as $V := (u, v) = (\dot{x}, \dot{y})$, and the relationship between I_x, I_y, velocities and I_t are stated compactly in (14). Since the image domain is quantized, the necessary derivatives such as I_x, I_y and I_t can be computed numerically. With these definitions, we have:

$$I_t = -\left(I_x u, I_y v\right) = -\nabla I \cdot V. \tag{14}$$

Velocities in x and y directions (u, v) will be computed using iterative methods as there is only one equation but two unknowns (u, v). Horn and Schunk suggest loose classification technique which minimizes the general error and the noise error to solve this problem, (Horn & Brian, 1992). Partial derivatives can be calculated as given as in (15)–(17), and axial indexing to calculate them is shown in Fig. 3:

$$I_x \approx \frac{1}{4}\left(I_{i,j+1,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i+1,j,k} + I_{i,j+1,k+1} - I_{i,j,k+1} + I_{i+1,j+1,k+1} - I_{i+1,j,k+1}\right), \tag{15}$$

$$I_y \approx \frac{1}{4}\left(I_{i+1,j,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i,j+1,k} + I_{i+1,j,k+1} - I_{i,j,k+1} + I_{i+1,j+1,k+1} - I_{i,j+1,k+1}\right), \tag{16}$$

$$I_t \approx \frac{1}{4}\left(I_{i,j,k+1} - I_{i,j,k} + I_{i+1,j,k+1} - I_{i+1,j,k} + I_{i,j+1,k+1} - I_{i,j+1,k} + I_{i+1,j+1,k+1} - I_{i+1,j+1,k}\right), \tag{17}$$

where $I_{i,j,k}$ is grayscale density function for ith row jth column and kth temporal frame. i, j, and k index the entries in y, x and time directions, respectively. In order to determine the optical flow vectors accurately, illumination changes should be minimized using the smoothness measure given by (18), which is desired to be close to zero:

$$J_s^2 := \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2. \tag{18}$$

On the other hand, we also would like to maintain the quantity $J_b = I_t + \nabla I \cdot V$ around zero. This practically implies the minimization of the cost functional below:

$$J = \int\int_r (\alpha^2 J_s^2 + J_b^2)dxdy, \tag{19}$$

where α is a coefficient determining the relative importance of the terms contributing to the total cost defined over the image domain Υ. Defining $\mu(u)$ and $\mu(v)$ as the approximate Laplacians of the velocity components u and v, respectively, and using the calculus of variation for (19), we arrive at the following equations, which we can solve for u and v iteratively:

$$\left(\alpha^2 + I_x^2 + I_y^2\right)u = \left(\alpha^2 + I_y^2\right)\mu(u) - I_xI_y\mu(v) - I_xI_t, \tag{20}$$

$$\left(\alpha^2 + I_x^2 + I_y^2\right)v = -I_xI_y\mu(u) + \left(\alpha^2 + I_x^2\right)\mu(v) - I_yI_t. \tag{21}$$

Assuming velocity estimate for nth frame as (u^n, v^n), and $(n+1)$th frame as (u^{n+1}, v^{n+1}); velocities are computed using (22) and (23) iteratively:

$$u^{n+1} = \mu(u^n) - I_x\frac{I_x\mu(u^n) + I_y\mu(v^n) + I_t}{\alpha^2 + I_x^2 + I_y^2}, \tag{22}$$

$$v^{n+1} = \mu(v^n) - I_y\frac{I_x\mu(u^n) + I_y\mu(v^n) + I_t}{\alpha^2 + I_x^2 + I_y^2}. \tag{23}$$

The aforementioned process yields two velocity matrices having the same size with intensity image, say u and v, and the obtained velocity matrices will be used for obstacle avoidance and turn detection processes. For an in depth discussion of the Horn and Schunk method, the reader is referred to Horn and Brian (1992).

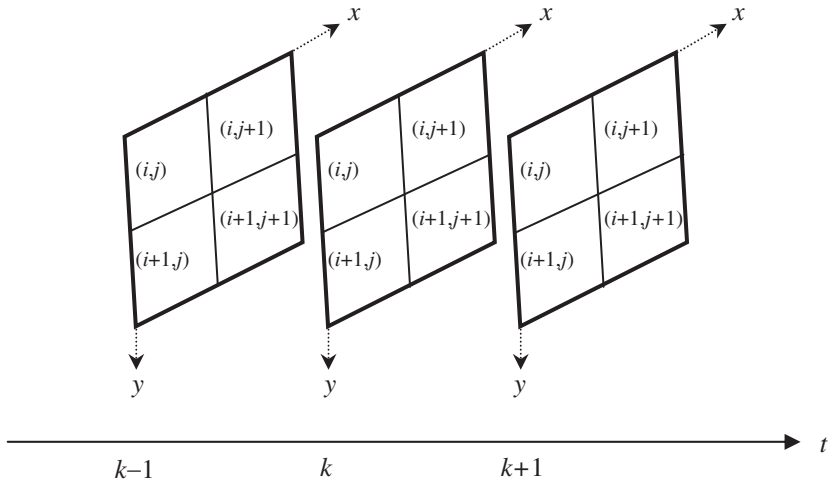

Fig. 3. Matrix which is used for calculating gradients x, y directions and time domain (k and $k+1$ are sequential video frames).

3.2. Obstacle avoidance using optical flow

Optical flow gives information about motion caused by the observer and objects, and this information makes it possible to detect obstacles and position of the objects in simple working environments. However, detection of obstacles is a tedious task in complex environments where there are typically low quality environment textures or noise on the frames. Hence, additional assumptions are inevitably imposed, such as computing time-to-contact, determining edges and corners to detect obstacles are just to name a few. Edge and corner information can be useful when the image is not corrupted by noise and the content is simple, however, as the operating conditions approach those of real life, edge detection may not be used effectively. Time-to-contact, or range information of obstacles, is commonly used in obstacle avoidance due to its usefulness. Nevertheless time-to-contact is affected by the structure and position of the obstacles. If the obstacle is close to the boundary of the image, calculated range information for obstacle is likely to be incorrect and UAV maneuvers based on such information is potentially dangerous.

It is known that magnitude of optical flow velocity for obstacles that are close is larger than those in the far. The proposed system here utilizes this fact to detect obstacles in a way seeking the minimum change in the optical flow to determine the allowed movement area in front of the quadrotor UAV. Running the straight motions and turning maneuvers at the junctions are principally different from each other. In the straight motion, the proposed algorithm discovers the most appropriate field to move in. The turning at a junction requires yaw axis motion and after detecting the obstacle free region, expected center of the optical flow velocity is determined to compute the reference yaw angle.

Required yaw angle for obstacle avoidance is determined by using the magnitude field of the optical flow which is obtained after resizing the grayscale image to the quarter of the original frame. This operation reduces the computational intensity of the optical flow calculation. Sum of the optical flow magnitudes are computed among various templates by running on the optical flow data in a predefined subimage instead of entire 2D data. The search window and template are represented in Fig. 4 where we see that the window slides along the horizontal axis but it covers only the rows in between 187 and 374, which are determined experimentally. After the horizontal scan along the shown strip (see Fig. 4), the center point of the template with a minimum sum of magnitudes is used to determine the reference yaw angle for the next control cycle. Denoting the width of the search window by S and

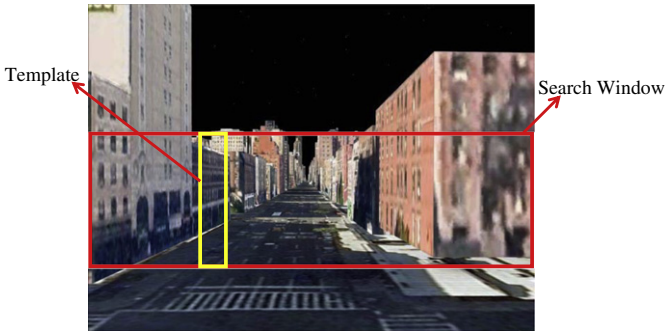

Fig. 4. Used search window and template in obstacle avoidance process.

that of the predefined template by T_ℓ, the necessary yaw command correction value (Ψ_{p1}) is determined as described by the pseudo code given in Table 2. The algorithm checks the center position of the template having the minimum sum of optical flow magnitudes and prescribes the required adjustment for the yaw angle to avoid obstacles.

In Table 2, $M(x, y)$ is the magnitude of optical flow for template point (x, y) and the pixel thresholds seen in the inequalities are the determined experimentally.

For the movements requiring turns, the algorithm determines the necessary second adjustment for yaw angle in addition to the first adjustment process by considering the expected value of the computed sum values. The expected value gives descriptive

Table 2
Pseudo code of obstacle avoidance for movements not requiring turns.

for $i = 1$ to $(S - T_\ell)$			
$T_s(i) = \sum_x \sum_y M(x, y)$			
end			
$i^* = \arg\min_i T_s(i)$			
Choose the template having the minimum sum of optical flow by setting $T_c = T_s(i^*)$			
Calculate the center of T_c and denote the center by C_c			
if	$C_c < 55$	then	$\psi_{p1} = -7°$
elseif	$55 \leqslant C_c < 65$	then	$\psi_{p1} = -3°$
elseif	$65 \leqslant C_c < 75$	then	$\psi_{p1} = -1.5°$
elseif	$75 \leqslant C_c \leqslant 85$	then	$\psi_{p1} = 0°$
elseif	$85 < C_c \leqslant 95$	then	$\psi_{p1} = 1.5°$
elseif	$95 < C_c \leqslant 105$	then	$\psi_{p1} = 3°$
elseif	$C_c > 105$	then	$\psi_{p1} = 7°$
Update the command signal for the yaw angle ($\psi_r \leftarrow \psi + \psi_{p1}$)			

Table 3
Obstacle avoidance for movements requiring turnings.

for $i = 1$ to $(S - T_\ell)$			
$T_s(i) = \sum_x \sum_y M(x, y)$			
end			
Calculate the expected value of T_s and denote the value by T_e which states the start position of the template			
Determine center position of the template which employs the expected value as center position and denote the center by C_e			
if	$C_e < 55$	then	$\psi_{p2} = 7°$
elseif	$55 \leqslant C_e < 65$	then	$\psi_{p2} = 3°$
elseif	$65 \leqslant C_e < 75$	then	$\psi_{p2} = 1.5°$
elseif	$75 \leqslant C_e \leqslant 85$	then	$\psi_{p2} = 0°$
elseif	$85 < C_e \leqslant 95$	then	$\psi_{p2} = -1.5°$
elseif	$95 < C_e \leqslant 105$	then	$\psi_{p2} = -3°$
elseif	$C_e > 105$	then	$\psi_{p2} = -7°$
Update the command signal for the yaw angle $(\psi_r \leftarrow \psi + \psi_{p1} + \psi_{p2})$			

information about free maneuvering area after the turning. The pseudo-code of the algorithm is presented in Table 3.

3.3. Detection of junctions with optical flow

Collision free navigation with a UAV like the one considered here is involved mostly with problems having a destination and autonomous flight is a typical requirement to plan the motion at different levels. Although the industrial applications consider the flight management under several task layers handling different functionalities, the goal here is to exploit the information contained in the optical flow to perform successful maneuvers whenever commanded. In this respect, typical drawbacks of the available methods are the inability to perceive the junctions, failure in handling the sizes of the roads and costly memory requirements of the complicated flight management platforms. In Duchon (2006), in order to guide the turning maneuvers, a perception based navigation algorithm is used to detect the openings on the left and right sides utilizing large decrease in the optical flow magnitudes. Emphasized also in Sarcinelli-Filho et al. (2002), Heinrich (2002), Huang et al. (2006), Sirinivasan et al. (1996), Egelhaaf and Kern (2002), Barron and Srinivasan (2006), the viewpoint of considering the regions with larger optical velocities as obstacles is the classical approach to detect the obstacles. We propose the use of classical approach for determining the junctions, which are not obstacles indeed. The lateral optical flow (optical flow on left and right sides) decreases fast when the vehicle approaches a junction, and it gets larger when the junction is passed and there are again buildings on the sides. Using only the information available in the optical flow of the left and right halves of the camera image, we propose a new technique to detect the junctions and schedule correct maneuvers.

Remember the optical flow velocity magnitudes are larger for closer objects and assume that u^n and v^n are the optical flow matrices for the nth frame in x and y directions. The temporal variation matrices are computed as given in (24):

$$\begin{aligned} Du^n &= u^{n+1} - u^n, \\ Dv^n &= v^{n+1} - v^n. \end{aligned} \tag{24}$$

After calculation of temporal variations in the optical flow, the gradient of these matrices ($G_x := \nabla Du^n, G_y := \nabla Dv^n$) are calculated. The matrices G_x and G_y are partitioned into two halves to represent the left and right of the acquired image:

$$\begin{aligned} G_{xL} &= G_x(r,c), \quad 1 \leqslant r \leqslant R, \quad 1 \leqslant c \leqslant C/2, \\ G_{xR} &= G_x(r,c), \quad 1 \leqslant r \leqslant R, \quad 1 + C/2 \leqslant c \leqslant C, \\ G_{yL} &= G_y(r,c), \quad 1 \leqslant r \leqslant R, \quad 1 \leqslant c \leqslant C/2, \\ G_{yR} &= G_y(r,c), \quad 1 \leqslant r \leqslant R, \quad 1 + C/2 \leqslant c \leqslant C, \end{aligned} \tag{25}$$

Table 4
Junction detection algorithm.

Calculate Dx_i, Dy_i			
Calculate G_x, G_y			
Partition to obtain $(G_{xL}, G_{xR}, G_{yL}, G_{yR})$ and recast them into a column vector			
Calculate υ_{xL}, υ_{xR}, υ_{yR}, υ_{yL} and υ_{mL}, υ_{mR}			
if	$(\upsilon_{mR}/\upsilon_{mL}) \geqslant 1$	then	ratio = $\upsilon_{mR}/\upsilon_{mL}$
elseif	$(\upsilon_{mL}/\upsilon_{mR}) > 1$	then	ratio = $\upsilon_{mL}/\upsilon_{mR}$
if	$(\upsilon_{mL} < T_L)$ and $(\upsilon_{mR} < T_L)$ and (ratio < 1.5)	then	C_R = true
else			C_R = false
if	C_R = true	then	Calculate Euclidean distances for three possible actions
Perform the action resulting in minimal Euclidean distance between the vehicle and the destination point			

where R and C stand for the total number of rows and columns of the image, which has been scaled down to a quarter of the original image. Obtained gradient matrices are resized into single column vectors, say X, and variances of these vectors are computed using $\sigma^2(X) = E[(X - E(X))^2]$ where X $\{\upsilon_{xL}, \upsilon_{xR}, \upsilon_{yR}, \upsilon_{yL}\}$. Then, magnitudes of the variances for left and right sides are calculated using (26). Achieved magnitudes for the left and right sides are utilized to determine the junction for proper navigation of the vehicle:

$$\begin{aligned} \upsilon_{mL} &= \sqrt{\upsilon_{xL}^2 + \upsilon_{yL}^2}, \\ \upsilon_{mR} &= \sqrt{\upsilon_{xR}^2 + \upsilon_{yR}^2}. \end{aligned} \tag{26}$$

If the vehicle chooses to move along an unusually wide street, the information available in the sides mislead the navigation scheme such that the vehicle assumes that it is approaching a junction although in reality there is not a junction ahead. In order to prevent such a misguidance, the quantity given by $\upsilon_{mR}/\upsilon_{mL}$ (or $\upsilon_{mL}/\upsilon_{mR}$) is checked. If $\upsilon_{mR}/\upsilon_{mL}$ is much larger than 1.5 the vehicle understands that there is not a junction and the building on the right is closer than that on the left. This enables to fly along wide streets without colliding with the buildings.

Variance magnitudes are continuously monitored and when both of them decrease under a predefined threshold value, denoted by T_L, the system understands that it is approaching a junction. When a junction is detected the vehicle decides on the best action among the three possible actions namely, turn right, turn left or continue the straight motion. The criterion is to get closer to the destination point and the Euclidean distance between the vehicle and the destination point is evaluated before implementing any of these actions. When the best action is determined, the vehicle is commanded to that direction. The algorithm guides the vehicle till it is one meter away from the destination point. The pseudo-code of the junction detection process is as given in Table 4.

4. Experiments in the virtual environment

The experiments considered here are realized in the Google Earth® environment. A dynamic link between the Matlab/Simulink® and Google Earth® is established to observe and perform realistic motions. To perform the autonomous flight experiments, we choose New York City (NYC) streets as the information available in NYC is as much detailed as we need. Latitude and longitude values required for global positioning are obtained from Google Earth® on start, and then the latitude and longitude values are updated using movements of the UAV. The presented experiments assume the following:

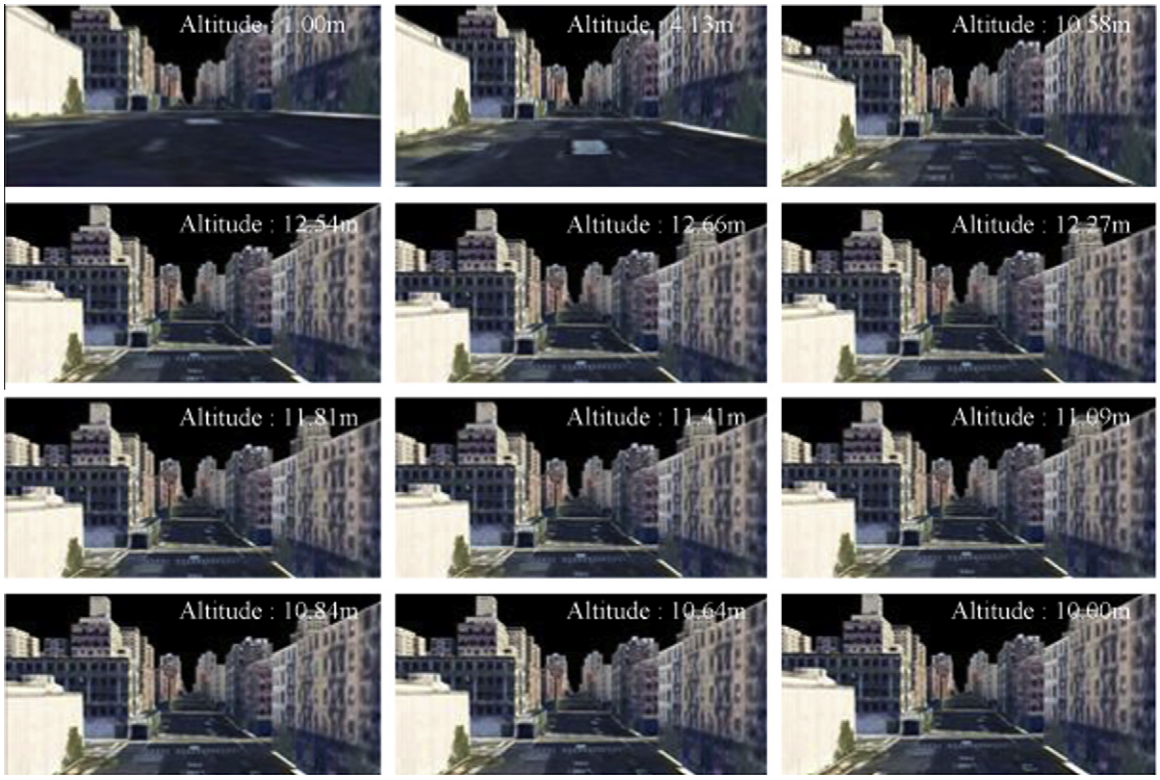

Fig. 5. Captured frames during hover.

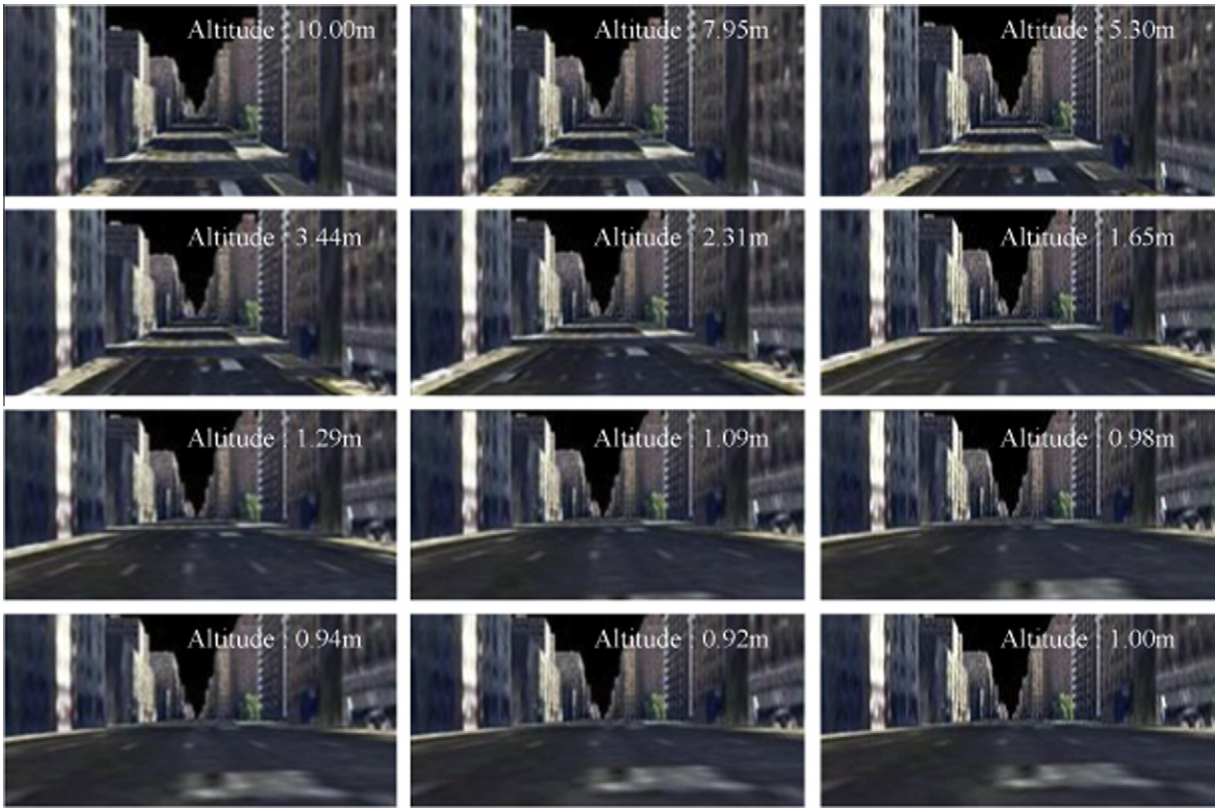

Fig. 6. Captured frames during landing.

(1) The trees are not obstacles as Google Earth® environment does not provide information about them.
(2) There are no air-specific disturbances such as winds, rain etc.
(3) Internet connection speed is assumed to be good enough to obtain the information about the buildings, i.e. buildings appear quickly in the scene.

In the proposed system, the first phase is the hover operation managed with the aid of PID controllers. The visual feedback is not utilized during this operation. Closed loop control of the altitude dynamics (z axis) is achieved by using the altitude data acquired from the Google Earth®. This keeps the quadrotor UAV at the desired heights, i.e. ten meters from the ground in this paper. Vision-based navigation of the UAV is started after the desired altitude is reached. A set of figures depicting the vehicle viewpoint during the hover are shown in Fig. 5 and those for landing are depicted in Fig. 6. The time interval between the consecutive figures is equal to 1 s, and the entire flight management algorithm is depicted in Fig. 7.

As seen in Fig. 8, white filled rectangle area indicates the maximally obstacle free area. Utilizing the center position of the area, which is denoted by C_c in Table 2, the first component of the desired yaw angle is computed and this is denoted by Ψ_{p1}. If no turns is needed, the desired yaw command is computed by $\Psi_r = \Psi + \Psi_{p1}$. For the cases where turns are required, an additional adjustment component denoted by Ψ_{p2} is computed using the expected value inferred from the sum of the optical flows evaluated along the strip shown in Fig. 4. If the expected center point, denoted by C_e in Table 3, arises in the middle of the frame then it is decided that the obstacles lie on both sides of the observer, and they are nearly the same distance away from the vehicle. If the expected center point is on the left side of the frame, the obstacles to be avoided are mostly in the left side of the frame, and if the center point is on the right side of the frame, obstacles are decided to be on the right side of the frame. Depending on the value of the variable C_e, Table 3 suggests an additive correction value for the yaw angle reference as described by $\Psi_r = \Psi + \Psi_{p1} + \Psi_{p2}$.

Sequential frames and the corresponding gradients of the optical flow matrices are illustrated respectively in the left and right subplots of Fig. 8. In the top row, there is a building seen on the right and the building is marked with a rectangle. In the subsequent frames, the building disappears slowly and the variance of the optical flow decreases on the region corresponding to the building. When the variance decreases under the selected threshold value, the system detects the junction (see $(n+11)$th–$(n+12)$th frames of Fig. 9).

In addition to the aforementioned discussion, the vehicle is not allowed to perform lateral movements as the visual information is provided only in the forward direction. The velocity of the vehicle in the straight motion is set to 1 m/s, and when a turn is realized, the forward velocity is set to zero and optical flow calculation is also stopped as the turning maneuvers bring highly difficult scenes to analyze. These issues necessitate relatively slower UAV motions in the Cartesian space and the choice of the quadrotor structure is deliberate due to this fact.

Four different destination points are selected to test the proposed scheme. Firstly, a distant destination point is selected to observe the behavior of the UAV. As seen in Fig. 10, the UAV successfully arrives at the destination point while avoiding the obstacles and identifying the junctions to perform necessary turning maneuvers. When the UAV reaches 10 m away from the destination point, the UAV averts its heading to the destination and proceeds to go ahead till it reaches the destination. This mode does not utilize the optical flow calculation as the optical flow information around the destination point is not descriptive for the destination itself. When the error falls under 1 m the vehicle lands automatically. The proposed system is executed for the other three destination points as well. The UAV successfully arrives at the specified destinations without colliding to any of the buildings. The autonomous flight results are shown in Figs. 10–14. The destination point in Fig. 10 and that in Fig. 14 are the same but the threshold values for the variances are different. As seen from these two figures, this changes the vehicle's ability to distinguish different routes having different roads with different widths. Variance value for the destinations shown in Figs. 11–14 exploit a threshold

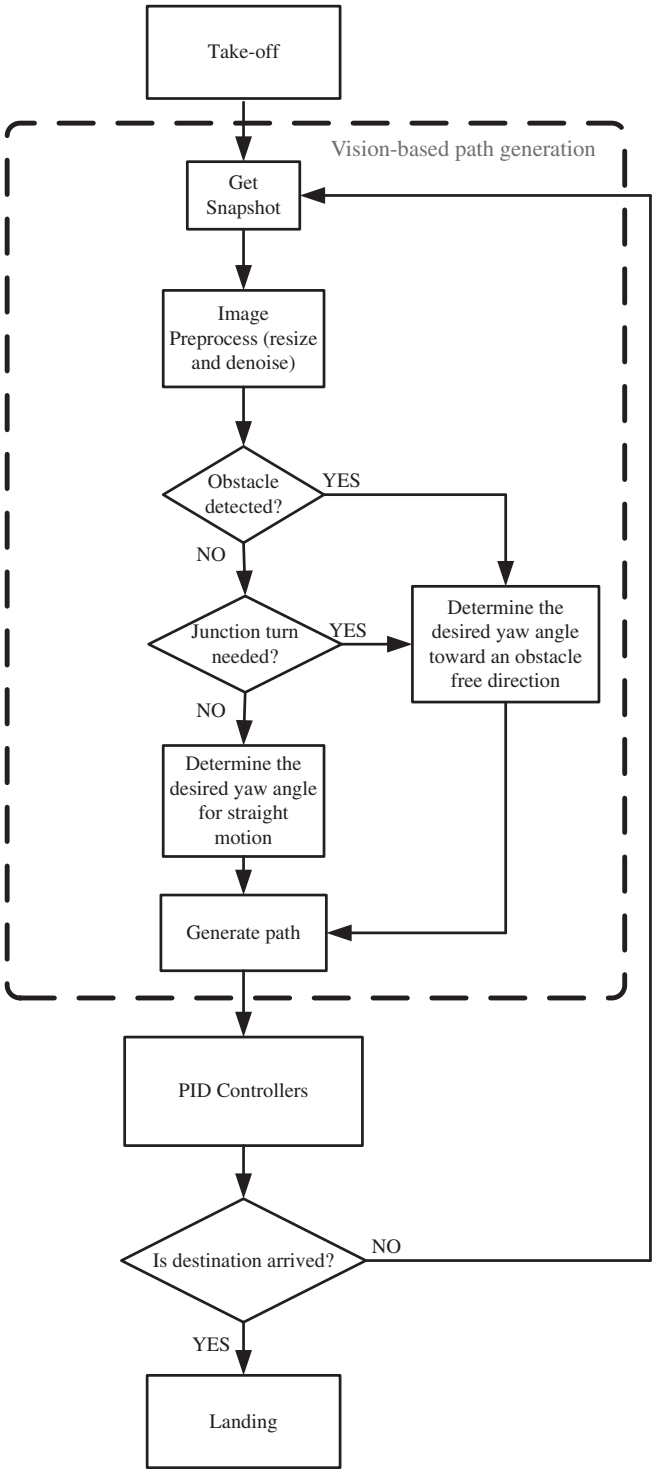

Fig. 7. Flow diagram of the proposed autonomous navigation system.

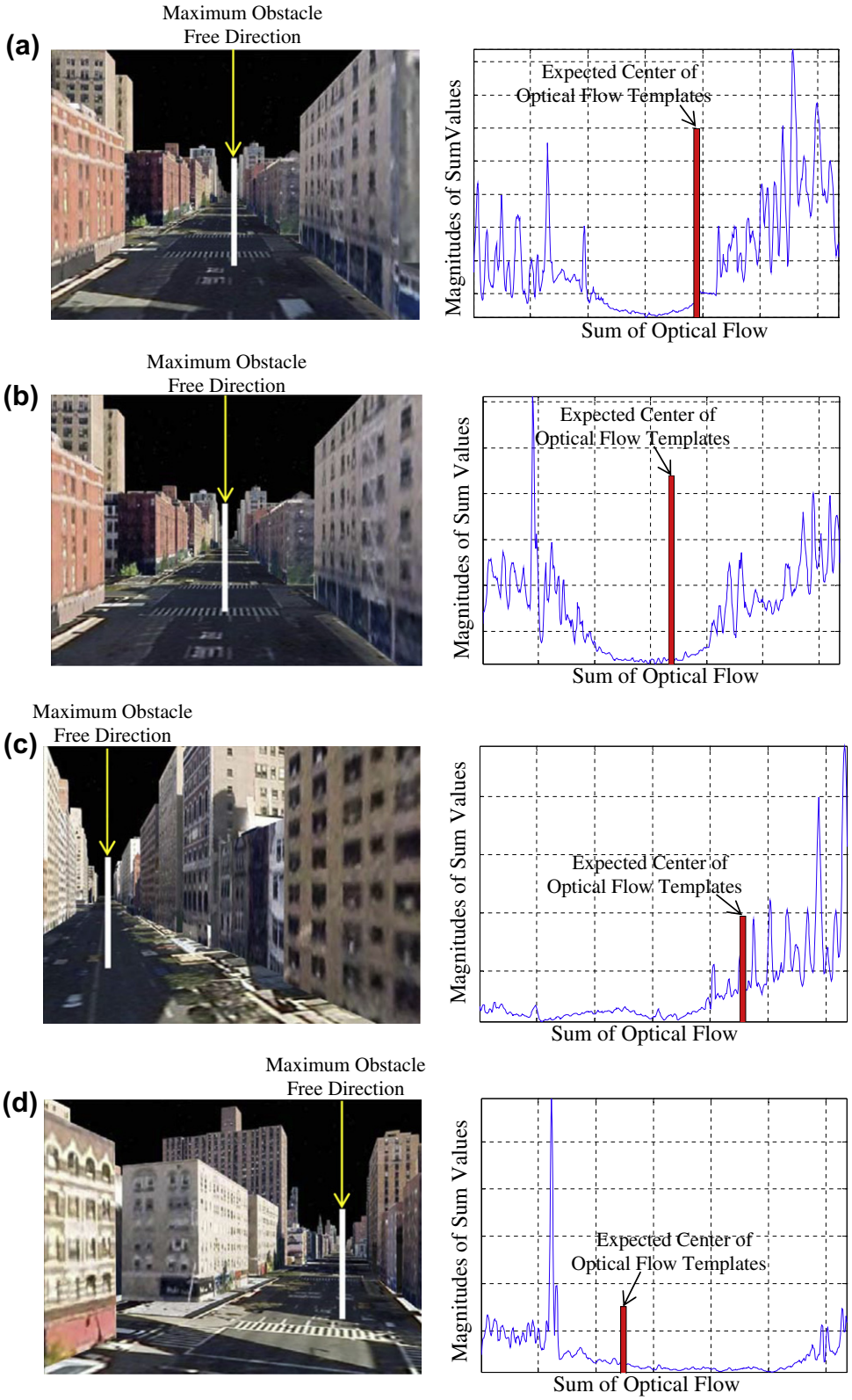

Fig. 8. Object detection process results (1st yaw adjustment are left sides, 2nd yaw adjustments are right sides).

value of 1.5 and it is clearly seen that the flight performance for this value is good for the roads having relatively smaller width. In the flight result seen in Fig. 12, the vehicle lands on a parking lot indicating that the vehicle is not restricted to land on destinations lying only on the streets.

Above discussion stipulates that the variance information is critical in achieving a collision free flight. More explicitly, choosing a large threshold value may result in ignorance of the obstacles and this may eventually cause a crash. The threshold values between acceptable levels, on the other hand, have the effect of changing

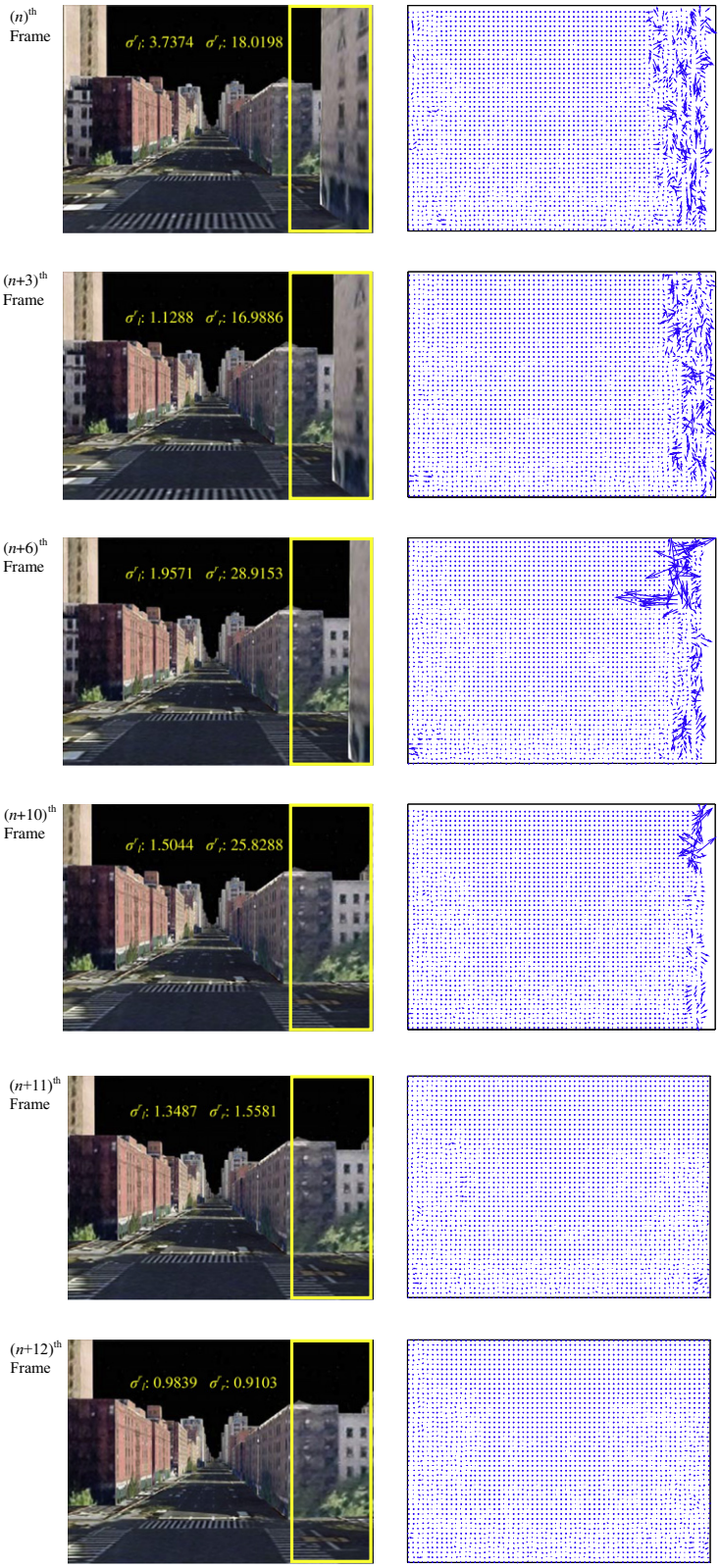

Fig. 9. Sequential frames and gradients of optical flow differences between sequential frames (while the building is disappearing variance value is decreasing under threshold value (1), and after $(n+12)$th frame is processed, crossroad detection notices to a junction) destination.

the route to be followed by the vehicle. It has been demonstrated experimentally that the smaller threshold values prevent detection of narrower junctions, middle level ones lead to the emergence of desired response and higher levels trigger the potential danger of collision. The existence of a best value depends on the availability of different width roads from the starting point to the destination.

Fig. 10. Result of the first experiment. Variance threshold is 0.5 and the vehicle cannot see narrow roads after the start position.

Fig. 11. Result of the second experiment. Variance threshold is 1.5 and a proper turn from a wide road to narrow road is achieved after the start position.

Fig. 12. Result of the third experiment. Variance threshold is 1.5 and a proper turn from a wide road to narrow road is achieved after the start position.

Fig. 13. Result of the fourth experiment. Variance threshold is 1.5 and a proper turn from a wide road to narrow road is achieved after the start position and a wide junction is passed successfully just before the destination.

Fig. 14. Result of the fifth experiment. Variance threshold is 1.5 and a proper turn from a wide road to a narrow road is achieved after the start position. The vehicle then turns right to a wide road to approach the destination.

5. Conclusions

This paper demonstrates the use of optical flow information for autonomous flight. A quadrotor type UAV is considered as it is a vertical takeoff and landing type vehicle. Its dynamic model has been realized in the Matlab/Simulink® environment and the attitude (the Euler angles) of the vehicle as well as the Cartesian dynamics of the UAV is equipped with a set of PID controllers. The reference signals to be tracked are derived utilizing the optical flow based flight management system. The virtual flight is performed in the Google Earth® environment and exemplar cases are considered in the streets of NYC. The flight is autonomous, i.e. no information about the path to be followed is provided to the vehicle, which is only provided the coordinates of the destination point. A link between the Matlab/Simulink® environment, which runs the dynamic model, and Google Earth® environment, where all high level information is contained, is established and the UAV is expected to hover, find its way to the destination and land in the vicinity of the destination point. Optical flow information is utilized in detecting the junctions as well as a comfortable forward motion is also maintained using the optical flow

technique. The problem in both cases is to derive the reference yaw angle to traverse an appropriate path. The constraint is to avoid the obstacles and the goal is to reach a predefined target point.

One of the key parameters is discovered to be the variance of the gradient of optical flow differences. The approach presented entails the selection of a threshold value for this parameter and different threshold values result in different paths. Assuming there are various paths with various widths in between the starting point and the destination, it is observed that smaller variance threshold values make the vehicle interpret small magnitude activities in the video as obstacles and it expects larger magnitude activities to detect a junction. Apparently, large magnitude changes are caused by wide streets and therefore for smaller threshold values, the vehicle cannot distinguish the junctions with narrow roads. On the other hand, larger variance threshold makes the vehicle detect both narrow and wide streets. This is due to the increased sensitivity to small magnitude changes in the scene. Despite slight increase of the threshold improve the visibility of narrow roads, further increase of the threshold may make the vehicle blind as the oversensitivity triggers incorrect response under circumstances where no junctions are seen and this may lead to eventual crash.

This shows that very small threshold values and very large threshold values make the vehicle blind and there exists an acceptable interval in which the autonomous flight response is satisfactory.

The contribution of the paper is to describe an autonomous flight scheme employing the optical flow information. Claims are validated in a Google Earth® linked Matlab/Simulink® environment. Future work of the authors aims to implement adaptive variance thresholding to increase the degree of autonomy.

Acknowledgment

This work is supported by TÜBİTAK 1001 Programme, Grant No. 107E137.

References

Amir, M., & Abbass, V. (2008). Modeling of quadrotor helicopter dynamics. In *International conference on smart manufacturing application* (pp. 100–105).

Ang, K. H., Chong, G., & Li, Y. (2005). PID control system analysis, design and technology. *IEEE Transactions Control Systems Technology, 13*(4), 559–576.

Aström, K. J., & Hagglund, T. (1995). *PID controllers: Theory, design and tuning*. USA: The Instrumentation, Systems and Automation Society (ISA).

Barron, J. L., Fleet, D. J., & Beauchemin, S. S. (1994). Performance of optical flow techniques. *International Journal of Computer Vision, 12*(1), 43–77.

Barron, A., & Srinivasan, M. V. (2006). Visual regulation of ground speed and headwind compensation in freely flying honey bees. *The Journal of Experimental Biology, 209*, 978–984.

Bouabdallah, S., & Siegwart, R. (2007). *Design and control of quadrotors with application to autonomous flying*. Ph.D. Thesis. Ecole Polytechnique Federale De Lausanne.

Braillon, C., Pradalier, C., Crowley, J.L., & Laugier, C. (2006). Real-time moving obstacle detection using optical flow models. In *Intelligent vehicles symposium* (pp. 466–471).

Call, B., Beard, R., & Taylor, C. (2006). Obstacle avoidance for unmanned air vehicles using image feature tracking. In *American institute of aeronautics and astronautics guidance, navigation, and control conference and exhibit*.

Camus, T. A. (1997). Real-time quantized optical flow. *The Journal of Real-Time Imaging, 3*, 71–86.

Castillo, P., Lozano, R., & Dzul, A. (2005). *Modeling and control of mini-flying machines*. London, USA: Springer-Verlag.

Chen, M. (2003). *Formation and flight control of affordable quadrotor unmanned aerial vehicles*. Ph.D. Thesis. The University of British Columbia.

Dev, A., Kröse, B., & Groen, F. (1997). Navigation of a mobile robot on the temporal development of the optic flow. In *Proceedings IEEE international conference on intelligent robots and systems* (pp. 558–563).

Duchon, A. P. (2006). Maze navigation using optical flow. In P. Maes, M. Mataric, J. A. Meyer, J. Pollack, & S. Wilson (Eds.), *Proceedings of the international conference of adaptive behavior. From Animals to Animats* (Vol. 4, pp. 224–232). Cambridge, MA: MIT Press/Bradford Books.

Egelhaaf, M., & Kern, R. (2002). Vision in flying insects. *Current Opinion Neurobiology, 12*(6), 699–706.

Fleet, D. J., & Jepson, A. D. (1990). Computation of component image velocity from local phase information. *International Journal of Computer Vision, 5*(1), 77–104.

Heinrich, S. (2002). Fast obstacle detection using flow/depth constraint. In *Proceedings of the IEEE intelligent vehicle symposium*.

Horn, B. K. P., & Brian, G. S. (1992). Determining optical flow. *Artificial Intelligence, 17*, 185–204.

Huang, W. H., Fajen, B. R., Fink, J. R., & Warren, W. H. (2006). Visual navigation and obstacle avoidance using a steering potential function. *Robotics and Autonomous Systems, 54*(4), 288–299.

KaewTraKulPong, P., & Bowden, R. (2001). An improved adaptive background mixture model for real-time tracking with shadow detection. In *Proceedings of 2nd European workshop on advanced video based surveillance systems*.

Klappstein, J., Vaudrey, T., Rabe, C., Wedel, A., & Klette, R. (2009). Moving object segmentation using optical flow and depth information. In *Proceedings of the 3rd Pacific Rim symposium on advances in image and video technology* (pp. 611–623).

Kurnaz, S., Cetin, O., & Kaynak, O. (2010). Adaptive neuro-fuzzy inference system based autonomous flight control of unmanned air vehicles. *Expert Systems with Applications, 37*(2), 1229–1234.

Lu, N., Wang, J., Yang, L., & Wu, H. (2007). Motion detection based on accumulative optical flow and double background filtering. In *Proceedings of the world congress on engineering* (Vol. 1, pp. 602–607).

Lucas, B. D. (1984). *Generalized image matching by the method of differences*. Ph.D. Thesis. Carnegie-Mellon University.

Lucas, B., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of imaging understanding workshop* (pp. 121–130).

Sarcinelli-Filho, M., Schneebeli, H. A., Calderia, E. M. O., & Soria, C. M. (2002). Optical flow-based obstacle detection and avoidance in mobile robot navigation. In *Proceedings of the 10th IEEE Mediterranean conference on control and automation*.

Sirinivasan, M. V., Zhang, S. W., Lehrer, M., & Collett, T. S. (1996). Honeybee navigation en route to the goal: Visual flight control and odometry. *The Journal of Experimental Biology, 199*(1), 237–244.

Spagnolo, P., Leo, M., D'Orazio, T., Caroppo, A., and Martiriggiano, T. (2006). An energy-based background modelling algorithm for motion detection. In *Proceedings of the 3rd international conference on informatics in control, automation and robotics, robotics and automation* (pp. 378–383).

Watanabe, Y., Johnson, E., and Calise, A. J. (2005). Vision-based approach to obstacle avoidance. In *American institute of aeronautics and astronautics guidance, navigation, and control conference and exhibit*.

Zhang, S., Yao, H., and Liu, S. (2008). Dynamic background subtraction based on local dependency histogram. In *The 8th international workshop on visual surveillance*.