Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 8937176, 11 pages
http://dx.doi.org/10.1155/2016/8937176

Research Article

Vision Based Displacement Detection for Stabilized UAV Control on Cloud Server

Hyeok-June Jeong,[1] Jeong Dan Choi,[2] and Young-Guk Ha[1]

[1] *Department of Computer Science and Engineering, Konkuk University, Neungdong-Ro, Gwangin-Gu, Seoul 143-701, Republic of Korea*
[2] *Smart Mobility Research Group ETRI, 218 Gajeong-ro, Yuseong-gu, Daejeon 34129, Republic of Korea*

Correspondence should be addressed to Young-Guk Ha; ygha@konkuk.ac.kr

Received 4 June 2016; Accepted 28 July 2016

Academic Editor: Nik Bessis

Nowadays, image processing solution is used in many fields such as traffic information systems and illegal intrusion detection systems. Now, to assist with the control of camera-equipped devices, appropriate image processing techniques are needed for moving rather than fixed observers. For achieving this goal, an algorithm should derive the desired results quickly and accurately; thus, this paper considers two characteristics: functional performance (reliability) and temporal performance (efficiency). Reliability means how well the desired results can be achieved, and efficiency means how quickly the result can be calculated. This paper suggests an optimized real-time image algorithm based on the integration of the optical flow and Speeded-Up Robust Features (SURF) algorithms. This algorithm determines horizontal or vertical movement of the camera and then extracts its displacement. The proposed algorithm can be used to stabilize an Unmanned Aerial Vehicle (UAV) in situations where it is drifting due to inertia and external forces, like wind, in parallel. The proposed algorithm is efficient in achieving drift stabilization by movement detection; however, it is not appropriate for image processing in small UAVs. To solve this problem, this study proposes an image processing method that uses a high-performance computer.

1. Introduction

Images taken from the camera have a lot of information. These images may subsequently be processed with techniques that are used very successfully in various applications, such as systems for intruder detection, car accident detection, and license plate recognition. These systems make our lives convenient in many different ways; however, these applications are not addressed in this research because these systems use real-time images taken at a fixed location, such as a CCTV camera.

Recently, there has been a growing trend of using image processing techniques with images from cameras mounted on devices such as UAVs, robots, and automobiles [1]. It has the same effect as that of the device that has the ability to look around. In particular, it is sometimes hard to obtain information from the inertial or infrared sensors of UAVs or mobile robots. In these situations, real-time image processing can be a very powerful solution.

However, image processing techniques should be used carefully in fixed camera situations, because they pose a risk of unexpected events. Furthermore, incorrect image processing can lead to seriously hazardous situations. In order to eliminate these risks and ensure its reliability, an algorithm should be designed to deal with a number of risky cases and to use sophisticated methods for preventing mistakes; however, these precautions lead to a great increase in an algorithm's operation time. Image processing algorithms require a proper balance between the two factors, as calculation time is considered to be a very important factor in real-time image processing.

This paper presents an image processing algorithm to detect the movement of camera-equipped devices and estimate their displacement; it presents an optimized method to achieve this goal effectively and analyzes the effectiveness and reliability of the algorithm.

This paper proposes a method that controls a UAV so that it can fly in a very stable hovering pattern using the visual information recorded by a mounted camera. There are several image processing techniques that are used to achieve these goals but this paper proposes a method that works with cloud computing in order to apply this technology to small UAVs. This control concept is called "cloud robotics" and is an efficient and reliable system [2].

2. Related Work

There have been many prominent studies that have aimed to achieve drift stabilization. To solve the stabilization problem, [3] proposed a system for detecting displacement using an inertial sensor. This control system showed excellent results in Matlab Simulink; however, the processing system based on integrated sensor data is not working in the real world as implemented in the simulation. In fact, it is very difficult to measure motion information using the inertial sensor if a UAV is moving at constant speed. It is possible to design a control system [4] that can stabilize drift using GPS, but this control system is useless if the GPS sensor is unable to receive signals (indoors, underground, in a tunnel, etc.).

There have also been many recent research projects in computer vision and image processing for UAV control. Our work in this paper is closely related to the works of [2, 5] and those that present a control method for a UAV tracking system, but these algorithms work on the computer on the UAV. They are not efficient for a micro-UAV because of its mass and energy consumption. In addition, the on-board computer system has poor performance compared with ordinary computers, which makes it more difficult to build a smarter control system. [6, 7] propose a way to control a UAV using cloud computing-based vision, but the papers do not describe practical control methods.

Research [8] has proposed a control system for stabilization which monitors and commands using vision information from a camera that is installed in the platform. This means that the control system works only on the Virtual Robotics Experimental Platform (V-REP). This paper eliminates the weaknesses in recent papers, achieving drift stabilization using real-time image processing and calculating the displacement of a UAV with cloud computing.

H.264 video coding technology is used in this paper for real-time video transmission [9]. A related paper [10] proposes an algorithm that tracks the displacement of its edges by extracting prominent features and rotating the image using the extracted displacement or movement. Another related paper [11] describes an algorithm that calculates quadrotor movement to infer a change in the edge. Though quite similar in topic to this paper, the algorithm in paper [11] performs poorly in detecting movement and is not optimized. Related paper [12] is similar to research paper [11].

Another related papers system [13] determines movement up to the current state based on the characteristics of the past with an edge extraction-based method. [14] proposes a concept for tracking camera motion using MPEG video. The algorithm in [14] is similar to ours but does not extract displacement information and is not suitable for use in real time. Some papers present algorithms that detect object movement and determine displacement using image processing with a camera.

All prior research shows excellent performance; however, these systems need to be improved because they do not have proper algorithms to rapidly calculate drift information to control UAVs. For this reason, the algorithm in this paper is expected to be an important contribution to UAV drift stabilization.

3. Overall Architecture

This paper will be described with a top-down approach; the first section presents the required top-level architecture diagram for the proposed system.

3.1. Top-Level Architecture of System. It is undesirable to mount high-performance computers that can perform image processing on small UAVs for several important reasons. First, the physical space is insufficient. Second, a small UAV cannot lift the computer; the total weight of the vehicle increases and very significantly reduces the efficiency of the BLDC motor [15]. This causes the motor to run out of extra torque, which can be an obstacle in controlling the attitude of the UAV [16]. Third, the electrical power efficiency of the UAV would seriously decrease. Because all devices on a UAV consume power from the same battery, a computer that performs many operations and low-efficiency motors resulting from weight increase can speed up battery consumption. For these reasons, we claim that it is not possible to perform image processing on a small UAV. However, a smart UAV control system should use a large amount of visual information. This paper proposes a system that performs real-time image processing in a cloud system and controls a UAV according to these results. The top-level architecture of the proposed system is shown in Figure 1. The embedded system on a small UAV is described in the diagram on the right. It contains only the minimum essential control functions, such as receiving sensor data, calculating values, and controlling the motor via PID control [17]. The UAV has a camera module that acquires images and performs minimal compression for transfer. The UAV can send or receive the control packet and status packet. The camera module communicates with an embedded system by USART and shares information with a high-performance computer via WiFi. The high-performance computer is installed on the ground. The high-performance computer receives a stable supply of electricity and can be freely extended as necessary. The camera module can reliably transmit and receive data within a limited distance [18]. Visual information transmission and data packet transmission and reception issues are not considered in this paper, nor is a small UAV on-board control system considered in this paper, as we use a verified UAV platform that flies stably and has been designed to respond immediately to commands from the user. It is necessary to design a method that cooperates with the cloud computer because its efficiency is increased [19] by the development of algorithms (such as

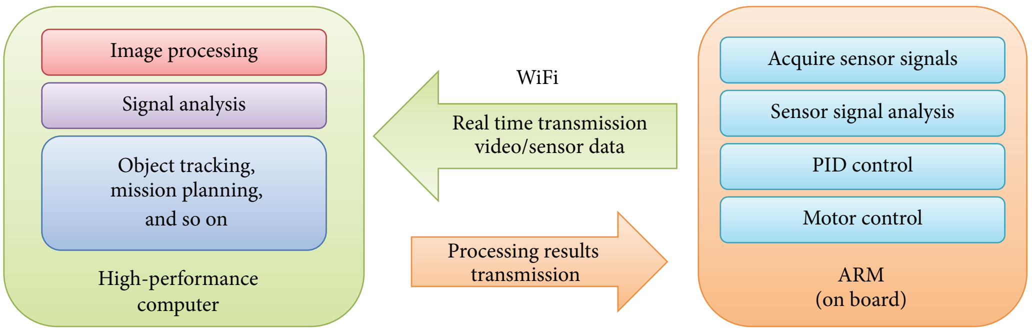


Image processing
Signal analysis
Object tracking, mission planning, and so on
High-performance computer
WiFi
Real time transmission video/sensor data
Processing results transmission
Acquire sensor signals
Sensor signal analysis
PID control
Motor control
ARM (on board)

FIGURE 1: The top level of architecture.

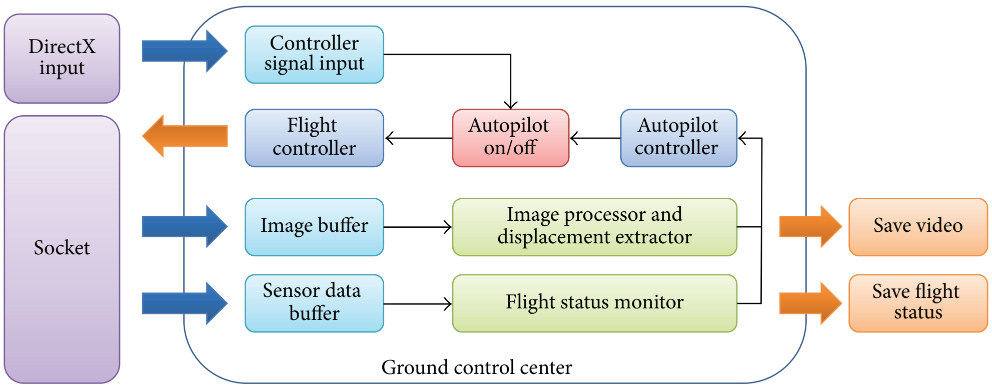


FIGURE 2: Architecture of ground control center.

H.264) and communication technology. The key idea of this paper is the design of a computer control system that operates on the ground. Its design is described in the next section.

3.2. Ground Control Center. This paper aims to control the UAV using the results real-time image processing on a cloud server. Ground control center is essential software that runs on the cloud server in order to achieve drift stabilization in a small UAV. The ground control center proposed is shown in Figure 2. Each block is represented in the following specific description:

(1) *DirectX Input*. It is a Windows DirectX library for receiving the joystick signal.

(2) *Socket*. It is TCP/IP-based communication for receiving real-time video and sensor data from the UAV and sending it a control packet.

(3) *Image Buffer*. The ground control center stores each image frame in a buffer as soon as it enters the socket.

(4) *Sensor Data Buffer*. The ground control center stores all sensor information in the buffer.

(5) *Image Processor and Displacement Extractor*. This procedure is a callback function that is invoked if one frame transfer is completed. The displacement is calculated using the Lucas-Kanade optical flow and Speeded-Up Robust Features (SURF) algorithms.

(6) *Flight Status Monitor*. This monitors the sensor data transmitted from the UAV, such as attitude information (pitch, roll, and yaw), battery voltage, air pressure, GPS, and infrared.

(7) *Autopilot Controller*. The controller performs drift stabilization using the displacement and sensor data. This procedure produces the control signal by the PID controller.

(8) *Controller Signal Input*. The input recalls the output signals required for flying from the joystick, such as axis information (pitch, roll, yaw, and throttle) and information about which joystick buttons are pressed.

(9) *Autopilot On/Off*. The Autopilot On/Off selector decides whether to send the joystick signals to the flight controllers directly or to send the joystick signal mixed with an autopilot control signal.

(10) *Flight Controller*. The controller generates 20 control packets per second. Each control packet consists of basic flight control information such as pitch, roll, yaw, throttle, and auxiliary control information for special purposes. The ground control center is designed to perform a complex and elaborate operation, so program reliability is of great importance. If there is a slight mistake in the receipt of a joystick signal, the transmission of a command packet, or the accuracy of image processing or Automatic Flight Control, a hazardous situation can occur. For this

reason, all program modules should be carefully designed to faithfully perform their roles. However, it is not possible to increase only the reliability of the system blindly because image processing (step (5) Image Processor and Displacement Extractor) requires a very large amount of computation. The system is designed to balance reliability and efficiency through optimization.

4. System Design

4.1. The Necessity of Integration: SURF and Optical Flow. Sensor data integration is useful because integration can exploit advantages of the combined systems, such as the uniform high-accuracy trajectory information of GPS and the short-term stability of inertial navigation systems (INS) [20]. A GPS receiver is a low frequency response navigation sensor that can provide instantaneous position accuracy at 1 Hz. INS provides continuous position, velocity, and orientation estimates, which are accurate in the short term but are subject to drift due to sensor drifts. Each piece of displacement information calculated by the SURF and optical flow algorithms shares properties with GPS and INS information. SURF provides uniform high-accuracy displacement information and optical flow can provide short-term displacement stability. However, displacement information calculated by SURF is noisy and optical flow cannot calculate absolute axis, since at any given time only one independent measurement is available from each image sequence. The integration of SURF and optical flow can limit the shortcomings of the individual systems, namely, the noise typical of SURF measurements and the long-term drift characteristics of optical flow. Integration can also exploit advantages of the two systems, such as the uniformly high-accuracy displacement information from SURF and the short-term stability of optical flow. Calculating velocity using optical flow is as follows.

4.2. Calculating Velocity Using Optical Flow. All organisms can recognize their movement by looking at the constant flow of surrounding objects or landscapes. This concept is adopted in our system to calculate the displacement of a small UAV. Optical flow can determine the vector that features flow for each image frame in real time.

4.2.1. Integral-Based Displacement Extraction. The displace vectors can be obtained by using optical flow [21]. That means the vectors represent camera motion. Paper [21] suggests following equation:

$$\begin{gathered}\mathrm{Sum}_x = \sum_{i=0}^{n} \left(\mathrm{vector}_i\right)_x, \\ \mathrm{Sum}_y = \sum_{i=0}^{n} \left(\mathrm{vector}_i\right)_y.\end{gathered} \tag{1}$$

Equation (1) means summation of all optical flow vectors on x-axis and y-axis. This technique makes it possible to obtain displacement. However, it has limitation for detecting exact displacement. This issue will be mentioned in the next chapter.

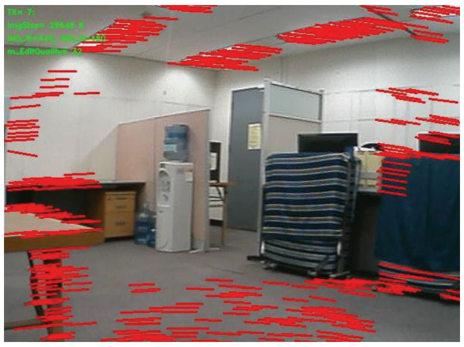
4.2.2. Operation of Optical Flow. The operation of the displacement detection system based on optical flow is presented in Figure 3. Figure 3(a) shows the zero vector at the stop state and holding only the feature points. Figure 3(b) shows vectors with components that represent movement. As previously explained, optical flow can provide short-term displacement stability. This means that the system can detect a stopped state, a moving state, and movement direction, as shown Figure 3(c). However, the final values do not converge to zero, despite the camera returning to the original position. That is, the algorithm cannot calculate the distance from the absolute reference position because it is specifically designed for extracting the flow of information between two images.

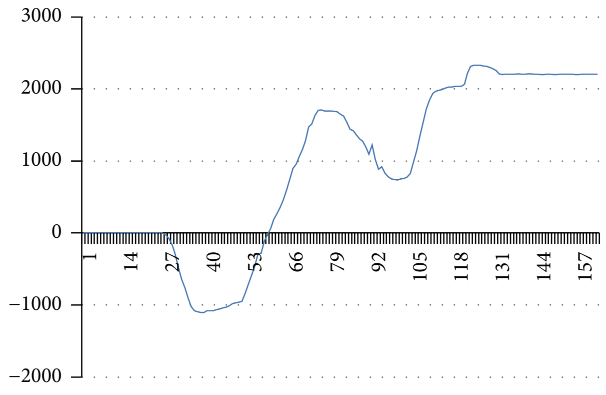
4.3. Absolute Distance: SURF. Another way to determine displacement is to recognize the movement of particular objects. Since this recognition is based on specific objects, the observer is able to gauge his or her absolute position if the object does not move suddenly. This recognition method was imitated so that computer programs can recognize movement. The SURF method was applied to this system; not only are SURFs detector and descriptor fast, but the former is also repeatable and the latter is distinctive. SURF is well-suited for object detection, object recognition, or image retrieval tasks [22]. It is necessary to modify the SURF algorithm in order to use it to calculate displacement. The algorithm analyzes feature points from two different image frames and then extracts vectors between the most similar two features from each image frame. It can calculate the displacement from a specific position if one image frame defines that position.

4.3.1. Architecture of SURF for Displacement. The SURF algorithm is widely used for object tracking by analyzing the similarity of features between two images. However, if the system generates a reference image at a specific position, it can obtain the displacement between the stream image (the current position) and the reference image (Figure 6). The calculated displacement can be considered the displacement of the camera. For this reason, the system receives the reference image at the location where it should be held, as shown in Figure 4. Since the reference image is generated once at the desired location, it is possible to calculate the displacement when the camera is moved. Its advantage over optical flow is its ability to determine and return to the initial position. That is, the solution can define the camera movement in an absolute coordinate system.

4.3.2. Optimization of Proposed SURF Algorithm. The proposed system has optimized SURF for calculating the displacement of a camera. The original SURF algorithm requires too much time for the purpose of this system because it extracts more feature points than needed for the displacement calculation. In other words, the core of the optimization process is to extract appropriate feature points, which can be achieved by adjusting the size of the input images. The time

(a) View of starting point


(b) Movement detection



(c) Graph of movement detection (optical flow)

Figure 3: Experiment result of optical flow.

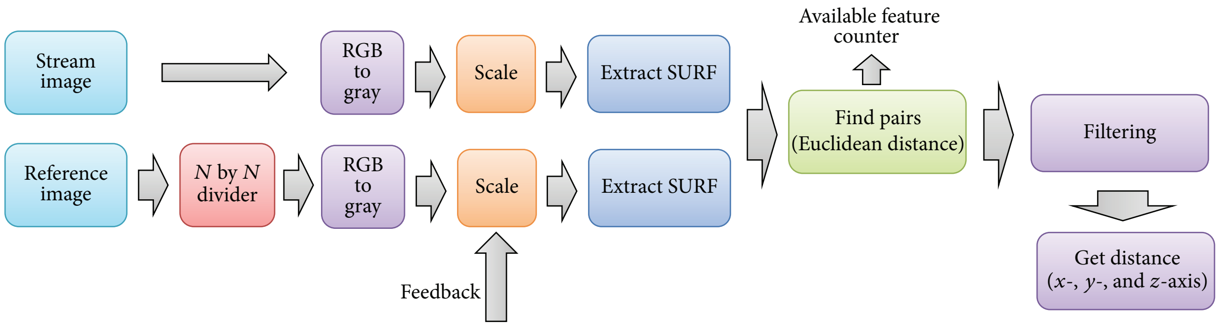


Figure 4: Architecture of SURF for displacement.

complexity of the original SURF algorithm is $O(n)$, and calculation time can be reduced to one-quarter of this by halving the images size (so that the number of pixels is reduced to 1/4). However, reducing or limiting the input size inflexibly can cause reliability problems when the system receives an image from which it is difficult to extract feature points; if the displacement from the reference image increases, the number of features that can be used for the displacement calculation is significantly reduced. For this reason, the system should be designed to find the appropriate number of features. The scaling factor for the image is determined by the number of extracted features, as shown in Figure 8. This scale is reduced when the number of extracted features is greater than the number of reference features. Similarly, the scale increases if the number of extracted features is less than the number of reference features. This concept is expressed as a time series [23]:

$$\begin{aligned}\text{width}(t) &= \text{width}(t) \times \text{scale}(t),\\ \text{height}(t) &= \text{height}(t) \times \text{scale}(t).\end{aligned} \tag{2}$$

Image resolution is determined by adjusting scaling of the height and width:

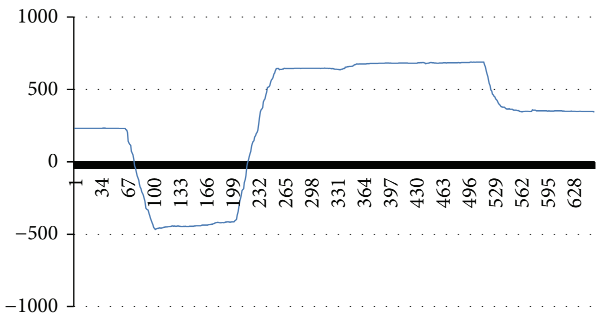
$$\text{scale}(t+1) = \text{scale}(t) + \left(\frac{\left(\text{Sum}_{\text{Features}}(t) - \text{Ref}_{\text{Features}}\right)}{1000}\right). \tag{3}$$

(a) View of starting point

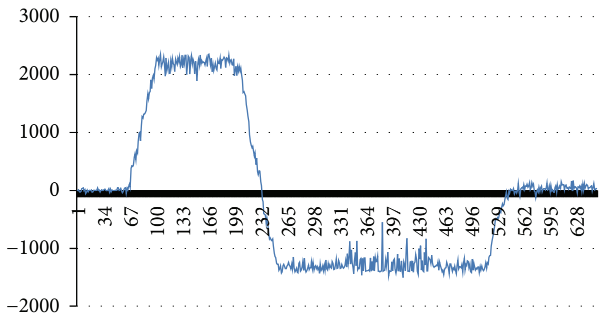
(b) Left movement detection

(c) Right movement detection

(d) Right movement detection (obstacle)



(e) Graph of movement detection by optical flow



(f) Graph of movement detection by SURF

Figure 5: Experiment result.

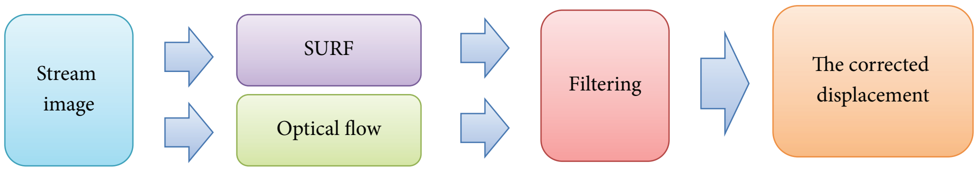


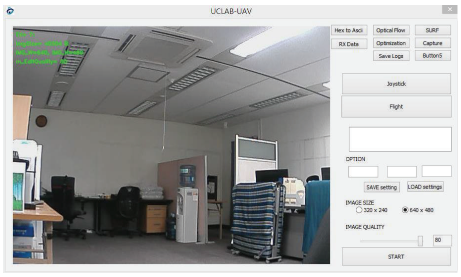
Figure 6: System implementation.

The following scale is adjusting to the new value, depending on the number of features present:

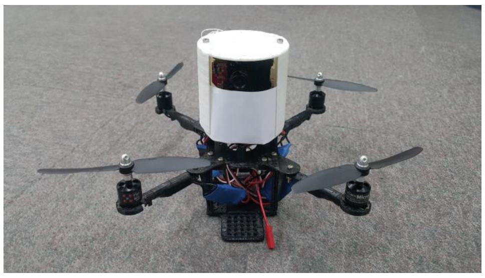
$$\min < \operatorname{scale}(t+1) < \operatorname{Max}. \tag{4}$$

The scale has a maximum and minimum value. According to (2)–(4), scale is determined flexibly according to the number of reference features. The performance and reliability required to calculate the displacement can increase by setting a flexible feature extraction algorithm.

4.3.3. Operation of Proposed SURF Algorithm. The operation of the SURF algorithm based on displacement extraction systems can be found in Figure 5. First, the system generates a reference image in a particular position, as shown in Figure 5(a). Displacement is calculated when the camera

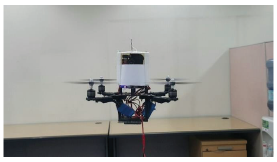


(a) 3D ground control center



(b) 3D printed quadrotor

Figure 7: System implementation.



(a) Quadrotor in flight in a lab

(b) Panoramic photo of the laboratory

Figure 8: Experiment result.

moves to the left, as shown in Figure 5(b), and the calculated displacement is shown in the Figure 5(f). When the camera is moved to the right, the vector display is shown in Figure 5(c) and the displacement values are shown in Figure 5(f). One of the benefits of the system is that the displacement can be extracted despite obstacles blocking the camera. Although the displacement is slightly unstable, it was calculated very well, as shown in Figure 5(d). Finally, the calculated displacement converges to zero when the camera returns to its starting position. The proposed system can accurately calculate the displacement of the system from a specific location, compared with optical flow, which cannot calculate an accurate displacement, as shown in Figure 5(e). However, the SURF algorithm is noisier than the optical flow algorithm. For this reason, we integrate the two types of data, SURF and optical flow, in order for the system to be more reliable.

4.4. Integration. The diagram of the proposed SURF/optical flow integration is shown in Figure 10. The functions of the blocks are as follows. Stream image: an image is received from the camera in real time. The frame rate is 30 FPS. However, the frame rate can be lowered based on WiFi reception. Once receipt of one image is complete, the following procedures are executed. SURF and optical flow: each image processing algorithm is executed independently using the same streamed image. The algorithms assign the result of the calculation and state the variables. Data filtering: this process is executed by the system when image processing is complete. This step is where integration occurs; the filter calculates the reliable data from the two complementary data sources.

4.4.1. Design Filtering Algorithm. In this paper, camera movement is limited in the x-axis (roll rotation). This limitation has been set because this paper aims to determine the validity of the proposed system. To add a y-axis calculation requires the addition of complex algorithms, such as depth cameras, and can be confusing to the system verification process. The system will be extended when the reliability and performance of the system are ensured:

$$\mathrm{MoV}_x(t) = \sum_{i=0}^{n} \left(\mathrm{vector}_i(t)\right)_x . \tag{5}$$

MoV (equation (5)) means Moment of Velocity. The velocity can be replaced with vector calculated by optical flow algorithms:

$$\mathrm{MA}_{\mathrm{MoV}_x}(t,n) = \frac{\mathrm{MoV}_x(t) + \cdots + \mathrm{MoV}_x(t-(n-1))}{n}. \tag{6}$$

A moving average (MA) is a calculation to analyze data points by creating a series of averages of different subsets of the full data set. The scope of the subset is defined in terms of the window size, which can be changed. Equation (6) calculates an average value of the latest of n MoV data points:

$$\mathrm{MA}_{\mathrm{SURF}_x}(t,w) = \frac{\mathrm{SURF}_x(t) + \cdots + \mathrm{SURF}_x(t-(w-1))}{w}. \tag{7}$$

Large window size enhances the reliability of the data; however, it causes the system to react slowly and follows the source data set badly. This means that a fixed window size can be an obstacle to calculating displacement. For this reason, the window size needs to be adjusted appropriately:

$$\frac{1}{w} = \mathrm{MA}_{\mathrm{MoV}_x}(t,n), \quad w! = 0. \tag{8}$$

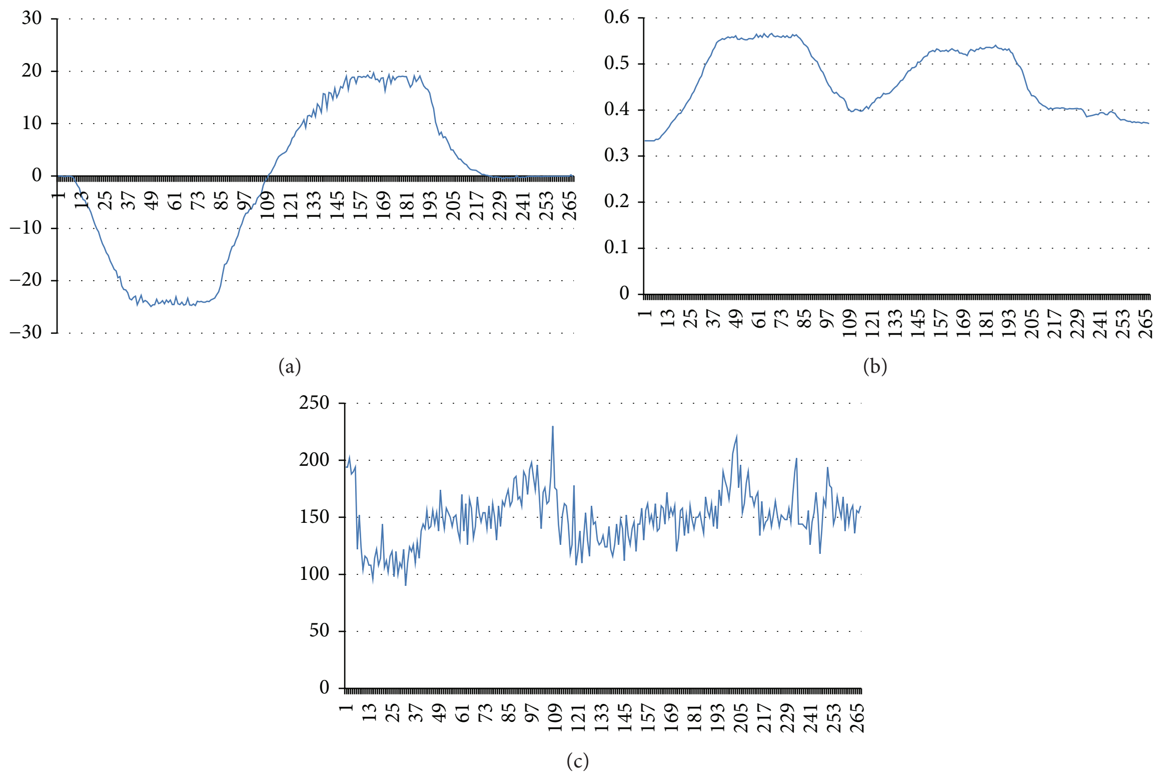

(a) (b) (c)

Figure 9: Architecture of SURF for displacement.

The window size has a boundary to enhance the reliability of the filtered data:

$$\text{Advanced SURF}(t, n) = \text{MA}_{\text{SURF}_x}\left(t, \frac{1}{\text{ABS}\left(\text{MA}_{\text{MoV}_x}(t, n)\right)}\right). \quad (9)$$

w is the window size for calculating the MA of SURF data and can be obtained from (8). That is, the final displacement is the MA value of the displacement obtained by the SURF algorithm, which has a flexible window size based on the MA value of the velocity of displacement.

To summarize the proposed equations (5) to (9): the system increases window size to improve reactivity when the velocity is large and it decreases window size to improve data stability when the velocity is small.

5. Implementation and Experiment

The proposed system is implemented using the MFC language, Visual Studio 2008 on a Windows 8 system. The system uses Open CV 2.3 for image processing.

The drone uses stm32-M3 as a main processor and 3D-printed frames. The drones thrust is generated using 4 BLDC motors and its attitude is controlled with an acceleration sensor, gyrosensor, and magnetic sensor. It is designed to control pitch, roll, yaw, and thrust by analyzing instruction packets transmitted through WiFi.

The experiment was carried out in a laboratory (Figure 7). As the laboratory is indoors, GPS reception was not available, so the system was drift stabilized by measuring displacement in the x-axis. The laboratory is an indoor space with white walls and some furniture, which means that the result features used by optical flow and SURF are not valid exclusively for this space.

The experiment was carried out in the environment shown in Table 1. The system was made for a PC of average performance.

6. Result

6.1. Result of Optimized Optical Flow and SURF. There are experimental results analyzing the performance and reliability of each of the optimization algorithms. Table 2 shows the accuracy and calculation time of the optimized optical flow algorithm. The proposed system shows 34 times improvement in performance over the unoptimized algorithm.

In addition, the proposed system shows equivalent results for the movement direction extracted from each algorithm. Although it does not sense low-pass filter, both systems show 100% movement direction detection.

Figure 9 shows the graph plotted from the SURF algorithm data optimized to calculate displacement measurements. Figure 9(a) shows displacement over time. First, the drone moves to the left, holds its position, and then moves to the right again. It is observed that it returned to the original point. Figure 9(c) shows that the scale adjusts to appropriately maintain 150 feature points.

Table 1: Results of the experiment.

	Computer 1	Computer 2
CPU	Intel I7 (Ivy) 3.4 GHz	Intel I5 (Ivy) @ 4.5 GHz
RAM	8 GB (PC1666)	24 GB (PC1666)
HDD	180 GB SSD	240 GB SSD
Operation system	Windows 8	Windows 8
Development tools	Visual Studio 2008 Keil 4	Visual Studio 2008 Keil 4
Library	Open CV 2.3 DirectX SDK 2006	Open CV 2.3 DirectX SDK 2006

Table 2: Results of the experiment.

	Proposed system	Normal systems
320 × 240	166FPS96.2% (6.02 ms) (100%)	83FPS94.2% (12.05 ms) (100%)
640 × 480	48FPS93.9% (20.83 ms) (100%)	11FPS 95.7% (90.91 ms) (100%)
Edge	400 (almost)	400 (fixed)
Dead zone	36% (fixed)	0%
Detected direction	Left side	Left side

Table 3: Results of the experiment.

	Time	Features (variance)	Scale (variance)
1	139.75 ms	762.44 (15199)	1 (fixed)
0.67	58.45 ms	453.26 (6046)	0.67 (fixed)
0.50	31.31 ms	260.46 (2900)	0.5 (fixed)
Advanced (150)	26.48 ms	149.85 (574)	0.45 (0.005)
Advanced (250)	40.31 ms	242.55 (781)	0.53 (0.011)

Table 3 shows the quality and reliability of the SURF algorithm optimized for displacement extraction. In the first three experiments, the scales are fixed at 1, 0.67, and 0.5, respectively. In the other two experiments, the objective features are fixed at 150 and 250.

First, as the scale of the images becomes smaller, operational time decreases. However, the average number of extracted valid features also decreases proportionately. The reduced number of features causes decreased reliability.

The variation of the number of features needs to be closely observed. Large variation means that there is a big deviation in extracted features at each frame. In other words, while the UAV is placed around the standard position, more features than average are extracted, but when the displacement is larger, fewer features than average are extracted.

In this system, when the scale is 1, the operational time increases and more features than needed are extracted, which wastes system resources. When the scale is 0.5, operational time and features are reduced, but the deviation in features is too large and the measurable scale of displacement decreases.

However, the proposed algorithm, by adequately controlling the scale at every moment, is able to approximately maintain the objective number of features desired. It is shown that this decreases deviation and is available to stably extract features in the total range of displacements. The two properties are plotted in Figure 10; the graph shows extracted displacement when the drone moves right and left and returns. It shows the characteristics of SURF, which is noisy, and optical flow, which does not converge to the original point.

6.2. Result of Integration. The two different properties of data are plotted in Figure 10. It means extracted displacement, when the drone moves right and left and returns. It is shown that the characteristics of SURF, which has big noise, and optical flow, which is not, converged to original point.

A smooth graph is generally shown by using MA, but the result has large delays from the use of real data. However, the displacement calculated by the Advanced SURF algorithm compounds the qualities of the two sources of data. It is as stable as a normal average filter and extracts values that instantly follow the tendency of the real data. Specifically, from frame 390, rapid displacement occurs, and Advanced SURF is well adapted to the condition.

6.3. Drift Stabilization of UAV. The real quadrotor control was carried out with integrated displacements in this paper. Displacement extraction in the x-axis and drift stabilization were carried out.

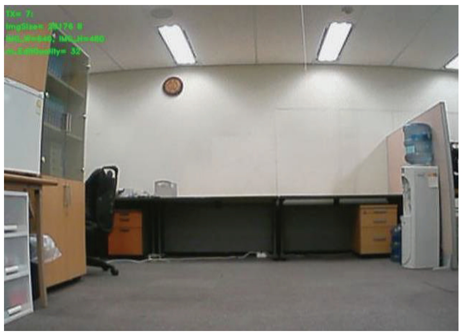
The displacement is measured from the standard point that the UAV is required to hold and is calculated by image processing. The measured displacement is input to the error value of the PID controller [24], and with the PID value, ground control center transmitted control packets to hold the position of the quadrotor. As shown in Figure 10, when the displacement is caused by flow, the quadrotor returns to the original point.

7. Conclusions and Future Work

This paper proposes a movement detection algorithm and a displacement measurement algorithm based on real-time image processing that integrates SURF and optical flow. The two image processing methods have been optimized for displacement detection. The proposed systems are designed based on feedback control to maintain optimal system performance and reliability. It is a powerful method for achieving the purpose of this research. As a result, our experiments show excellent functional and temporal image processing performance.

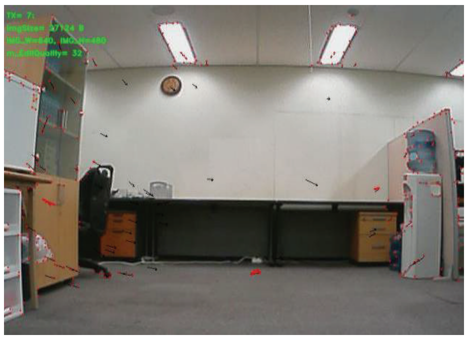
The proposed optical flow direction and velocity extraction algorithm has the accuracy of the original system, but its operation time is reduced to a quarter of that of the original. The Advanced SURF algorithm is 4 times faster than the original SURF while maintaining sufficient reliability when extracting displacement. The integrated displacement extraction algorithm is designed to maximize the advantages of SURF and optical flow, which are complementary data sources. It is a great contribution because it is a highly reliable, low-cost system.

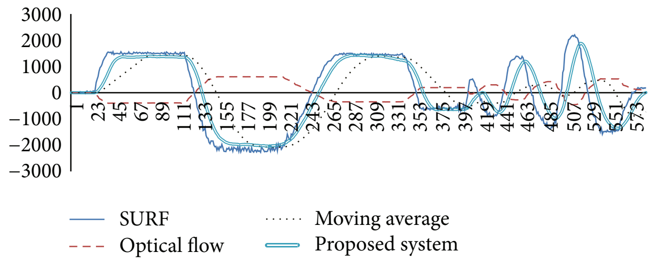
The reduction of processing time in real-time image processing allows it to run on a low-performance computer


(a) View of starting point

(b) Move right

(c) Move left


(d) Return to the start point



(e) Graph of SURF and optical and Advanced SURF

Figure 10: Experiment result.

and affords the ability to focus on adding safety modules or additional tasks to enhance performance. Finally, we created ground control center, executed on a cloud server based on the concept of cloud robotics; its performance and reliability are very high because it consists of optimized algorithms for stabilizing a UAV, such as the proposed image processing algorithms.

This system will be modified in the future. We are making another ground control center that can run on a smart device that has a 2 GHz CPU and greater than 2 Gb RAM. We anticipate that it will significantly reduce space restrictions because smart devices can move anywhere. The system also needs to be improved so that it can calculate displacement and control a UAV in another axis. These improvements will be achieved as our next research project. Another future research is the 3-axis motion recognition based on this result.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work was supported by the ICT R&D program of MSIP/IITP [R-20150224-000386, A Development of Software Technology for the Generation of Detailed Map Progressively Updated and the Driving Situation Awareness based on Cloud Computing]. This paper was written as part of Konkuk University's research support program for its faculty on sabbatical leave in 2013.

References

[1] S.-J. Han and J. Choi, "Real-time precision vehicle localization using numerical maps," *ETRI Journal*, vol. 36, no. 6, pp. 968–978, 2014.

[2] G. Hu, W. P. Tay, and Y. Wen, "Cloud robotics: architecture, challenges and applications," *IEEE Network*, vol. 26, no. 3, pp. 21–28, 2012.

[3] H.-J. Jeong, M. Hwang, H. Jung, and Y.-G. Ha, "Mathematical modeling of a multilayered drift-stabilization method for

Micro-UAVs using inertial navigation unit sensor," *Journal of Applied Mathematics*, vol. 2014, Article ID 747134, 11 pages, 2014.

[4] S. Lange, N. Sunderhauf, and P. Protzel, "A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments," in *Proceedings of the International Conference on Advanced Robotics (ICAR '09)*, pp. 1–6, IEEE, June 2009.

[5] T. Vladimir, D.-H. Kim, Y.-G. Ha, and D. Jeon, "Fast multi-line detection and tracking with CUDA for vision-based UAV autopilot," in *Proceedings of the 8th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS '14)*, pp. 96–101, IEEE, Birmingham, UK, July 2014.

[6] J. Chudoba, M. Saska, T. Baca, and L. Preucil, "Localization and stabilization of micro aerial vehicles based on visual features tracking," in *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS '14)*, pp. 611–616, May 2014.

[7] H. Guanglin, G. Rujun, and Y. Shi, "Application of FPGA in small UAV autopilot based on embedded LINUX system," in *Proceedings of the 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON '07)*, pp. 731–734, Taipei, Taiwan, November 2007.

[8] M. A. Olivares-Mendez, S. Kannan, and H. Voos, "Setting up a testbed for UAV vision based control using V-REP & ROS: a case study on aerial visual inspection," in *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS '14)*, pp. 447–458, IEEE, Orlando, Fla, USA, May 2014.

[9] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[10] J. K. Paik, Y. C. Park, and D. W. Kim, "An adaptive motion decision system for digital image stabilizer based on edge pattern matching," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 3, pp. 607–616, 1992.

[11] S. G. Fowers, D.-J. Lee, B. J. Tippetts, K. D. Lillywhite, A. W. Dennis, and J. K. Archibald, "Vision aided stabilization and the development of a quad-rotor micro UAV," in *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA '07)*, pp. 143–148, Jacksonville, Fla, USA, June 2007.

[12] M. Agrawal and K. Konolige, "Real-time localization in outdoor environments using stereo vision and inexpensive GPS," in *Proceedings of the 18th International Conference on Pattern Recognition (ICPR '06)*, vol. 3, pp. 1063–1068, IEEE, Hong Kong, August 2006.

[13] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV '03)*, pp. 1403–1410, IEEE, 2003.

[14] R. Wang and T. Huang, "Fast camera motion analysis in MPEG domain," in *Proceedings of the International Conference on Image Processing (ICIP '99)*, vol. 3, pp. 691–694, Kobe, Japan, October 1999.

[15] S. J. Park, H. W. Park, M. H. Lee, and F. Harashima, "A new approach for minimum-torque-ripple maximum-efficiency control of BLDC motor," *IEEE Transactions on Industrial Electronics*, vol. 47, no. 1, pp. 109–114, 2000.

[16] I. Takahashi and T. Noguchi, "A new quick-response and high-efficiency control strategy of an induction motor," *IEEE Transactions on Industry Applications*, vol. 22, no. 5, pp. 820–827, 1986.

[17] B. R. Trilaksono, R. Triadhitama, W. Adiprawita, A. Wibowo, and A. Sreenatha, "Hardware-in-the-loop simulation for visual target tracking of octorotor UAV," *Aircraft Engineering and Aerospace Technology*, vol. 83, no. 6, pp. 407–419, 2011.

[18] VARRAM, *Wireless Camera for Smart Device Datasheet*, VARRAM System Company, 2014, http://www.varram.com/.

[19] G. Ermacora, A. Toma, S. Rosa et al., "A cloud based service for management and planning of autonomous UAV missions in smart city scenarios," in *Modelling and Simulation for Autonomous Systems: First International Workshop, MESAS 2014, Rome, Italy, May 5-6, 2014, Revised Selected Papers*, vol. 8906 of *Lecture Notes in Computer Science*, pp. 20–26, Springer, 2014.

[20] H. Qi and J. B. Moore, "Direct Kalman filtering approach for GPS/INS integration," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 2, pp. 687–693, 2002.

[21] H.-J. Jeong and Y.-G. Ha, *Real-Time Vision Based Camera Moving and Displacement Detection Algorithm*, Computer Science and Its Applications, Springer, Berlin, Germany, 2015.

[22] S. Baker and I. Matthews, "Lucas-Kanade 20 years on: a unifying framework," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.

[23] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: speeded up robust features," in *Computer Vision—ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds., vol. 3951 of *Lecture Notes in Computer Science*, pp. 404–417, Springer, Berlin, Germany, 2006.

[24] K. H. Ang, G. Chong, and Y. Li, "PID control system analysis, design, and technology," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 559–576, 2005.