On Avoiding Moving Objects for Indoor Autonomous Quadrotors

Donghyeok Shin[1], Junwhan Kim[2], Byunggu Yu, and Dong H. Jeong[3]

***Abstract*— A mini quadrotor can be used in many applications, such as indoor airborne surveillance, payload delivery, and warehouse monitoring. In these applications, vision-based autonomous navigation is one of the most interesting research topics because precise navigation can be implemented based on vision analysis. However, pixel-based vision analysis approaches require a high-powered computer, which is inappropriate to be attached to a small indoor quadrotor. This paper proposes a method called the Motion-vector-based Moving Objects Detection. This method detects and avoids obstacles using stereo motion vectors instead of individual pixels, thereby substantially reducing the data processing requirement. Although this method can also be used in the avoidance of stationary obstacles by taking into account the ego-motion of the quadrotor, this paper primarily focuses on providing our empirical verification on the real-time avoidance of moving objects.**

I. Introduction

The use of Unmanned Aerial Vehicles (UAVs) has prompted much research for autonomous reconnaissance and surveillance. To perform any of such tasks effectively, a quadrotor is required to meet certain maneuverability required by the task. One of popular research topics on developing such a quadrotor is on how to make the quadrotor as autonomous as possible while satisfying the task's requirements [1]. Not only can it help minimize human pilot's mistakes, but the navigation can also be continuously optimized to improve the task's cost efficiency and the likelihood of success [2].

Autonomous navigation requires continuous knowledge about stationary and non-stationary obstacles (or moving objects) in the navigation space. In order to build a fully autonomous quadrotor, there have been many efforts on the detection of obstacles using laser range finders, artificial markers, and computer vision techniques [3]. A utilization of autonomous mini quadrotors is often considered for indoor airborne surveillance, payload delivery, and warehouse monitoring [4]. Indoor navigation environments pose major challenges associated with limited navigation space and unforeseen moving obstacles. Vision analysis techniques can provide high-accuracy obstacle representations. Although the high computation overhead has hindered the adoption of this approach in the auto-navigation of indoor mini quadrotors [5], the recent technological advances in the on-board-scale micro-computers, such as *Raspberry Pi*, *Intel Edison*, and *Arduino*, have been significant [6], and the distribution of advanced vision analysis techniques in open source libraries, such as OpenCV [7], has been accelerated. In the light of these recent developments, we focus on designing a vision-based method for detecting and avoiding obstacles for real-time autonomous navigation of mini quadrotors in indoor environments with moving obstacles.

In vision-based autonomous navigation, video stream analysis techniques are often used for detecting obstacles. Video streams are transmitted after compression to reduce overhead. Video compression methods, such as H.264, exploit similarities between adjacent frames and represent inter-frame deviations using compact motion vectors [8]. Traditionally, such motion vectors have received relatively less attention in autonomous navigation due their low spatial resolutions and relatively high noise levels. The noise can be reduced by means of signal smoothing filters (e.g., extended or unscented Kalman filter methods [9], [10]). In the area of vision-based autonomous navigation, more attention has been given to the high spatial resolution of the pixel-based representation rather than the temporal resolution that can be significantly increased by the motion-vector-based compaction. However, we believe that in a space-tight indoor navigation environments with dynamic moving obstacles, high temporal resolution is a key for real-time obstacle avoidance as long as the identification of available navigation paths is not compromised by the reduced spatial resolution.

In this paper, we present our study on Motion-vector-based Moving Object Detection (MMOD). The motivation behind this MMOD study is to show that one can exploit motion vectors to develop a robust and safe auto-navigation on a mini quadrotor platform for indoor navigation. A *stereo camera* is exploited to detect moving objects and their distances. The computational analysis of measuring motion-vectors and determining the distances to the detected objects is performed on a *Raspberry Pi* platform. Controlling a non-stationary airborne quadrotor in indoor environments is also managed automatically. To the best of our knowledge, our work is unique in terms of autonomously avoiding moving objects in real-time by utilizing vision-based motion vectors.

The rest of this paper is organized as follows. We overview related efforts in Section II. In Sections III and IV, we explain our proposed method and empirical results, respectively. The paper is concluded with Section V.

II. Related Work

Traditionally, object detection has involved intensive image processing. In vision-based object detection, a clear separation between background and foreground is considered

[1] Mr. Shin is a graduate student of the computer science and information technology (CSIT) department at the University of the District of Columbia (UDC), Washington, DC 20008, donghyeok.shin@udc.edu

[2] Dr Kim is with Faculty of the CSIT department at UDC, Washington, DC 20008, junwhan.kim@udc.edu

[3] Drs Yu, and Jeong are with Clearton, LLC (www.clearton.com), {byu, djeong}@clearton.com

as an essential step. Javed *et al*. [11] listed several major challenges in vision-based object detection as detecting quick illumination changes, understanding changes in background, and identifying unintentionally added foreground objects. They proposed a solution to detect objects by utilizing different processing methods on pixel, region, and frame levels. Such image processing method often requires high computational power. To lighten the computational load, often image resolutions need to be compromised to become a small scale (under 320 by 240) and temporal resolution have to be maintained below a real-time standard (below 20 frames per second), or use only partial regions sampled by interesting features of objects [12]. Since there is an additional requirement of supporting quick maneuvering for quadrotors, supporting both image processing and maneuvering in real-time is an extreme challenge.

To address the issue of supporting real-time processing, Hulens *et al*. [13] conducted several experiments with different small computing models to find optimal image-processing solutions in on-board computer to a quadrotor. More specifically, they tested different benchmark algorithms on eight different platforms and compared their performance results with the result from a desktop environment. As a requirement, the computers should be able to process 640 by 480 images at the speed of 10 frames per second. However, they found that about half of the models did not satisfy the requirement. From the study, it is known that a powerful computer is always required to support highly responsive quadrotor maneuvering if image processing is applied.

For delicate indoor flight, laser based sensing equipment can be loaded as a practical complement for image analysis. Wang *et al*. [14] proposed a method of utilizing a pre-manufactured scanning device to scan indoor environments. Since the equipment supports scanning 270 degrees viewing angle for 30 meters, scanned information is used to support a reliable maneuvering solution.

Image analysis heavily relies on pixel-by-pixel data. Since the size of the data can easily become too large, it is difficult to support real-time processing [15]. Alternatively, motion-vectors can be considered to support real-time processing because the data size is relatively small compared to the pixel-by-pixel data [16]. In vision-based autonomous navigation for UAVS, reducing the computational power is critical since it is closely connected to selecting an appropriate computer with a proper power supply. UAVS' movement speed can be decreased depending on the size of the computer and power supply.

Rodriguez-Canosa *et al*. [16] recognized the usefulness of motion vectors and provided a good criteria for separating foreground object motions from background motions. However, the experiment was conducted in an outdoor environment using a top-down camera. The maneuvering experiment used a pre-recorded video and focused on identifying exact shapes of objects using high-powered computer.

Kendoul *et al*. [17] worked on using motion as the important element for maneuvering the quadrotor. They conducted an experiment in a large indoor area to propose a GPS-independent technique. Their approach is somewhat similar to our proposed method. However, they conducted the vision-based computation on the separated desktop computer, and further exploitation on motion vectors to detect and avoid an object was not discussed. Although motion vectors is important for supporting real-time autonomous navigation for quadrotors, most researchers did not realize the importance of utilizing the motion vectors.

III. On Avoiding Moving Objects using Motion Vectors

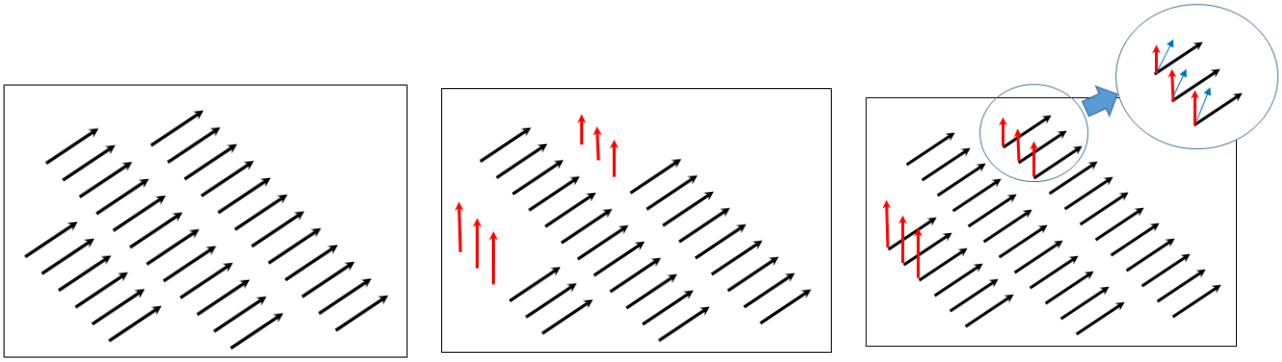
As described above, our method (i.e. MMOD) is to avoid moving objects in real-time. It is designed consisting of two steps - detecting and avoiding moving objects.

A. Detecting Moving Objects

A flying quadrotor is generating a continuous movement. Therefore, detecting moving objects is a major research challenge because motions can be created by the movement of the quadroter itself and any moving objects. To separate motions caused by the quadrotor and the moving objects, we defined the movement of the quadrotor as ego-motion (Q), and the movement of an object appeared in front of the quadrotor is considered as object-motion (O). At each frame, each grid of 16 by 16 pixels produces a motion vector. This vector can be categorized into $\overrightarrow{Q}$ and $\overrightarrow{O}$ generated by ego-motions and object-motions, respectively. Each vector consists of magnitudes and directions.

Figure 1 shows two different scenarios with having no motion and two motions. In Figure 1(b), two object-motions are detected. With the detected motions, $\overrightarrow{Q}$ and $\overrightarrow{O}$ are calculated on the magnitude and direction of motion vectors. It is important to note that background and foreground can be separated based on the obvious discrepancy between $\overrightarrow{Q}$ and $\overrightarrow{O}$. To identify the discrepancy, the mean and the deviations of motion vectors are used to filter out motion vectors with non-standard deviation of magnitudes and directions, which can be defined as a foreground (or stationary objects).

Using only motion vectors to identify background is too optimistic because a moving object may have multiple different motions. For example, a human's walking motion may be composed of multiple different motions of legs, arms, head, body, and so forth. This phenomena may lead to high error rate when generating motion vectors. To minimize this error rate, multiple *nearest* motions can be assumed as a single motion. Thus, a clustering algorithm is applied to MMOD for the combination of the nearest motions based on METIS [18]. A deterministic-threshold is defined as a minimum distance between motions, meaning that a cluster is at least the distance away from another cluster. Clustering threshold depends on the size of each quadrotor. Obviously, we do not need to know how many motions are in Figure 1(b) and even one of the motions in Figure 1(b) consists of how many motions are combined. However, a quadrotor must not fly toward the motions in order to avoid a collision.



(a) No Motion (b) Two Motions (c) Motion Estimation

Fig. 1. Detecting and Estimating $\overrightarrow{O}$ (a) consists of only motion vectors caused by $\overrightarrow{Q}$ because all motions have the same magnitudes and directions. Thus the motions are not real motions. In (a), we imply that the direction of $\overrightarrow{Q}$ is in the opposite direction of the motion vector. (b) represents objects (red arrows) generating two object-motions in two different areas. (c) indicates the superposition of $\overrightarrow{Q}$ and red arrows. The blue arrows of the enlarged circle indicates $\overrightarrow{O}$.

B. Avoiding Moving Objects

The second step of MMOD is to avoid moving objects after detecting the objects using motion vectors generated by captured video streams. Even though there are two moving objects are detected as Figure 1(b), the left motion is excluded from quadrotor's potential moving area because the motion does not belong to the quadrotor moving area. Figure 2 shows partitioned regions determined by computing object-motions. Once moving objects are identified ahead of the quadrotor, MMOD checks whether or not the moving objects belong to the quadrotor moving area.

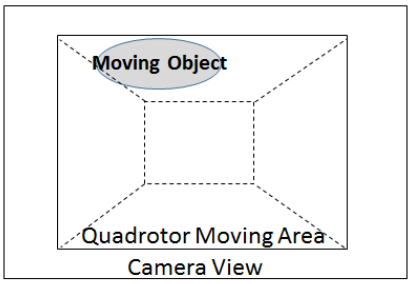


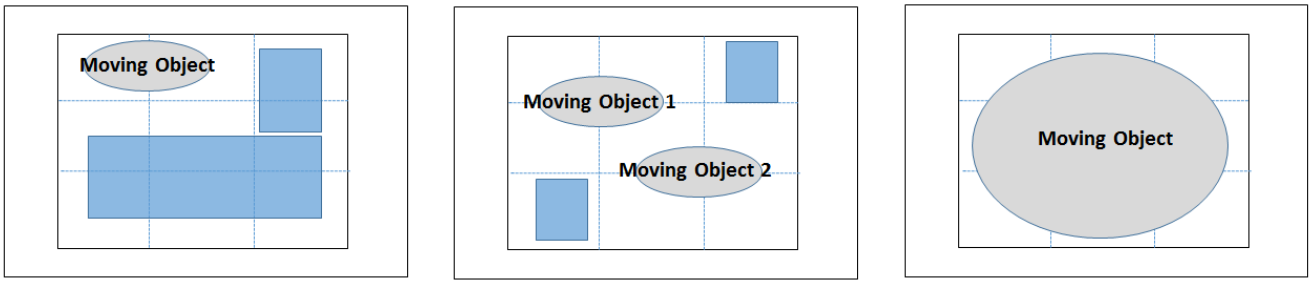
Fig. 2. Three Views of a Camera: The outer solid rectangular indicates the view of a camera, the inner solid rectangular represents the possible area where a quadrotor can move, the inner dotted rectangular shows what the quadrotor faces.

If a person keeps standing and moving only his/her arms, a quadrotor may fly into his/her body. If moving objects are temporally stopped, the corresponding motion vectors will not be generated. As another scenario, multiple moving objects which belong to the quadrotor moving area can be detected. To correctly avoid multiple moving objects, temporally stopped moving objects, and stationarily objects, a distance between such objects and the quadrotor must be determined to decide whether or not to change the direction of the quadrotor.

Recall that each grid of 16 by 16 pixels at each frame produces a motion vector. The detected motion vectors are categorized into $\overrightarrow{Q}$ and $\overrightarrow{O}$. Also a distance from the grid is calculated using an equation – $\frac{baseline \times focal\ length}{disparity} \times C$ [19], where a baseline is a length between two cameras, and C is a constant number. A disparity is defined as the difference in position between correspondence points in two images generated by a stereo camera. Unlike a stationary camera, the camera attached on the quadrotor is shaken by its ego-motion, so that it generates the same video but two different sets of motion vectors for each left and right camera. Such differences are utilized to derive the disparity. Even though $\overrightarrow{Q}$ is eliminated as a background to detect moving objects, $\overrightarrow{Q}$ is exploited to compute a distance between the quadrotor and background.

Figure 3 illustrates three different scenarios to identify moving objects and possible movable areas for the quadrotor. Since multiple movable areas can be detected, determining one area needs to be performed. Different priorities must be assigned to each area. Bottom area will have a lower priority because there are many objects on the floor in an indoor environment. Left and right sides are going to have the same priority. However, top area (i.e. ceiling region) maintain a higher priority because less number of obstacles will be in the air. Based on the priority information, if there are multiple objects are detected as shown in Figure 3(b), top area will be determined as a possible movable area. If no area is determined, the quadrotor will wait until at least one area is cleared.

The regulation is applied when objects are close to the quadrotor. In this paper, the maximum distance to analyze distances is empirically defined as 300 cm. A detailed explanation about how the distance is determined is included in Section IV. If the distance to the detected moving object cannot be determined, we assume that the object is located 300 cm away from the quadrotor. Thus, this regulation is feasible only when objects are detected and their distances are known.



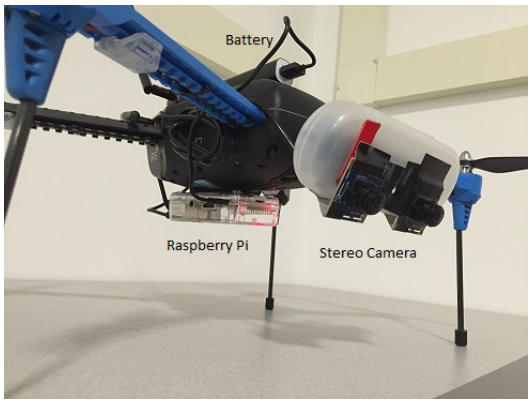
(a) A Moving Object (b) Two Moving Objects (c) A Large Moving Object

Fig. 3. Avoiding Moving Objects: (a) represents that a motion to be avoided is detected. Two blue transparent rectangles indicates possible ares where a quadrotor moves. If its distance is short, a turn right command is performed. Otherwise, it keeps going. (b) shows two motions and two possible areas to be moved. it gains altitude and move to right. (c) gives a motion but no possible areas.

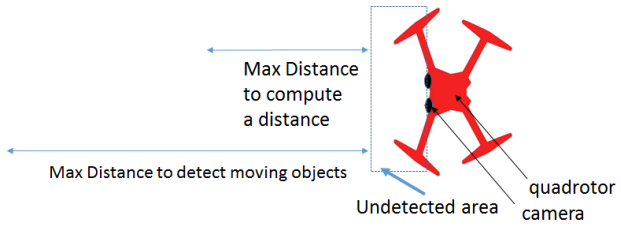
IV. Experimental Results

A. Our Quadrotor Platform

As a quadrotor, 3D Robotics IRIS+ (i.e., IRIS+) quadrotor shown in Figure 4(a) is used for the evaluation of MMOD. 3D Robotics IRIS+ has been known to be popular among many developers due to its origin of being open-source from the beginning. Raspberry Pi 2 B+ is built with 900 MHz quad-core ARM Cortex-A7, with Broadcom VideoCore IV GPU for H.264/MPEG-4 AVC high-profile decoding and encoding. MMOD composed of three threads is installed to Raspberry Pi. Each thread is in charge of capturing motion vector, analyzing motion vectos/distance, and navigating IRIS+, respectively. A Kalman filter [9] is applied to deduct noise generated by analyzing motion vectors and distances. Finally, a stereo camera equipped with 7 cm baseline is connected to Raspberry Pi linked to IRIS+.



(a) IRIS+ with Raspberry Pi and Stereo Camera



(b) A Top View of IRIS+ with Two Maximum Distances

Fig. 4. Our Quadrotor Platform to evaluate MMOD

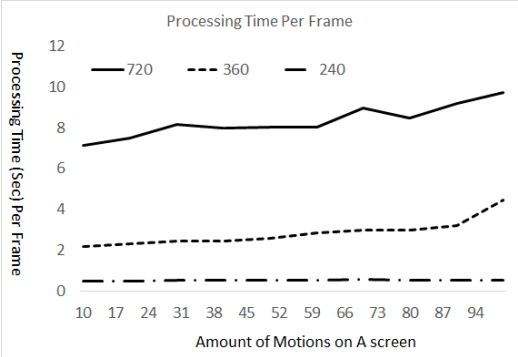
Based on a 7 cm static baseline, the maximum distance for objects to be calculated is about 300 cm. Also the maximum distance to detect moving objects is longer than that as illustrated in Figure 4(b). Thus, our quadrotor can detect moving objects and their distances at maximum 300 cm in front of it. Although a moving object more than 300 cm away is detected, its distance cannot be calculated. In the meanwhile, there is a blind spot between the quadrotor and the minimum detectable distance, called an "undetectable area", which is about 120 cm from the cameras, implying that a detectable area is only 180 cm. The detectable area defines the maximum speed of the quadrotor. If 180 cm per second as a speed and 30 frames per second as a temporal resolution (i.e., frame rate) are set, MMOD performs 30 commands while the quadrotor flies to 180 cm. The purpose of the evaluation is to show that the frame rate is more significant than the spatial resolution for automatous navigation.

To evaluate MMOD, we used a static baseline determining the maximum distance and the undetectable area within the minimum distance. On this configuration, MMOD controls IRIS+ using its programming protocol named MAVLink [20]. The protocol can adjust IRIS+'s altitude, pitch, roll, and yaw axis to change its directions, which include simple left and right.

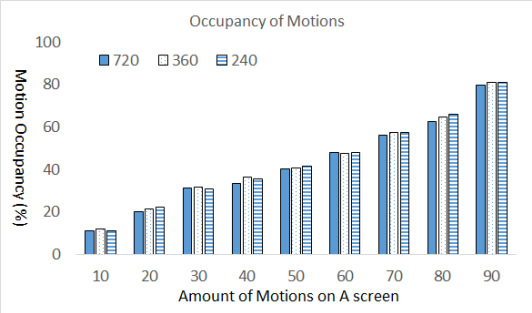
B. Evaluation Results

Before the evaluation to detect and avoid moving objects, we need to examine how fast a frame is analyzed due to limited navigation space. Motion vectors and distances are calculated for each frame. Obviously a high frame rate leads to accurate controls. In the meanwhile, an appropriate (spatial) resolution of a frame must be investigated. A microcomputer carried by the quadrotor has limited performance, so that high resolution degrades its performance.

Figure 5 shows the processing time and motion occupancy per frame among three resolutions – 720, 360, and 240 pixels with 30 frame rate – 30 frames per second. The processing time illustrated in Figure 5(a) represents how long Raspberry Pi takes to process a frame for the analysis of motion vectors. The motion occupancy shown in Figure 5(b) indicates how many $\overrightarrow{O}$s are detected on a frame. For example, Raspberry pi takes 7 seconds to process a frame on 720 pixels when



(a) Processing Time Per Frame



(b) Motion Occupancy On A Frame

Fig. 5. Processing Time vs. Motion Occupancy

20 % motions are detected. Due to its limited performance, the other frames are ignored. In this case, only a command is performed per 7 seconds. The quadrotor checks moving objects every flight distance 1,260 cm on assuming that 180 cm per second is set as a speed. Thus, 720 and 360 resolutions are not appropriate for auto-navigation.

As long as the amount of motions increases, the occupancy motion for three resolutions increases, but the resolution does not affect the amount of the detected motions. Regardless of the resolution, all motions are detected. Of course, the minimum resolution depends on the size of a quadrotor. In the meantime, the processing time per frame increases in the large amount of motions. For example, the processing times of 720 pixels is approximately 8 seconds on 50% motions. This implies that the quadrotor cannot perform any command for its navigation during 8 seconds if a 720 pixels resolution is selected. Thus, the low resolution (i.e., 240 pixels) is selected to accurately navigate the quadrotor due to the high processing time per frame.

To evaluate the effectiveness of MMOD, we consider two cases illustrated in Figures 6 and 7. Case I is to evaluate the identification of moving objects. Case II is for the evaluation of distances. For fair comparative studies, we tried to run the module of a traditional pixel-based vision analysis on our computer platform (i.e., Raspberry Pi), but it does not work properly due to computational overhead. Even though only two cases shown as Figures 6 and 7 are not adequate to generalize all scenarios using motion vectors, maximum two persons are involved into the angle and distance in which our quadrotor can detect, when assuming that only humans exist indoor as moving objects.

In Figure 6, a person (or a moving object) is moving ahead of a quadrotor. Figure 6(a) shows a camera view for reference. Based on the view, Figure 6(b) gives the set of motion vectors, and the superposition of the camera view and the set is illustrated in Figure 6(d) to check whether or not to fit into the motion vectors. Figure 6(d) shows the set of disparities on the camera view. The distance derived from the disparity is calculated, and then MMOD performs a turn-left command.

Unlike Case I, Figure 7 shows two moving objects. Two different groups of motion vectors are detected as shown in Figure 7(b). Figure 7(c) shows the superposition of the real moving objects and the groups. Even though two groups are distinct, there is no area for a quadrotor to be moved forward. However, the set of disparities is shown in Figure 7(d) gives also two different groups mapped with different gray colors, respectively. The bright color indicates a higher disparity, meaning that the corresponding object is closer to the quadrotor. As a result, the quadortor turns right.

V. Conclusion

In this paper, we present a new motion vector-based moving object detection method to support fully autonomous navigation for mini quadrotors. To show the effectiveness of our method, we tested it in an indoor environment. From the experiment, we identified that a quadrotor successfully avoids moving objects by directing itself towards the opposite side of the object. We also found that real-time autonomous moving objects avoidance is guaranteed since the temporal resolution is emphasised rather than the spatial resolution. We believe that our proposed approach is effective for avoiding moving objects without having any major time delay. In addition, the motion vector-based object detection we implemented in this study is a viable strategy for real-time autonomous navigation.

As mentioned above, our approach is evaluated in an indoor environment because of the uncertainty of outdoor environments. As a future direction, a robust outdoor autonomous navigation performing a compensation action when it moves toward an unwanted direction forced by wind and identifying the characteristics of moving objects can be introduced.

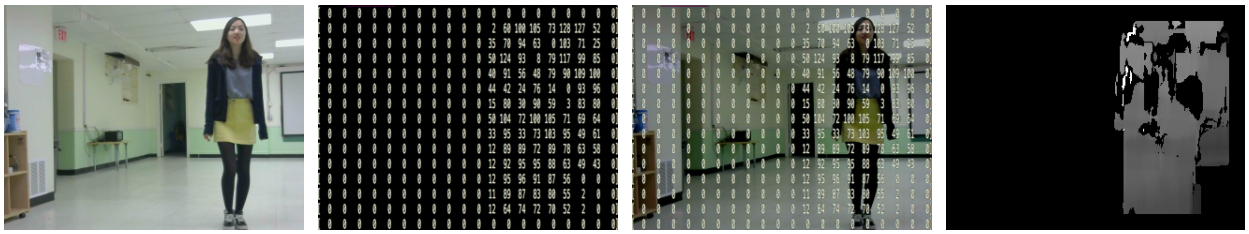
Acknowledgements

This work has been supported by US National Science Foundation STTR (Automation and Optimization of Aquaponics for Urban Food Production) under grant 1521153. The main purpose of this work is to monitor aquaponics installed in a greenhouse.

References

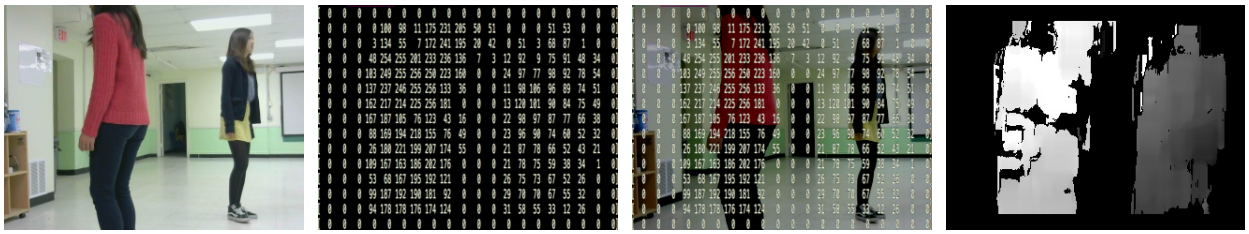
[1] A. Patel and S. Winberg, "Uav collision avoidance: A specific acceleration matching control approach," in *AFRICON, 2011*, Sept 2011, pp. 1–6.

[2] J. J. Engel, "Autonomous Camera-Based Navigation of a Quadrotor," *Masters Thesis in Informatik*, p. 103, 2011.



(a) Camera View (b) Motion Vectors (c) Image-based Motion Vectors (d) Disparities

Fig. 6. CASE I: A person is moving ahead of a quadrotor.



(a) Camera View (b) Motion Vectors (c) Image-based Motion Vectors (d) Disparities

Fig. 7. CASE II: Two persons are moving ahead of a quadrotor.

[3] Weiwei Kong, Daibing Zhang, Xun Wang, Zhiwen Xian, and Jianwei Zhang, "Autonomous landing of an UAV with a ground-based actuated infrared stereo vision system," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, no. 3, pp. 2963–2970, 2013.

[4] F. Mohammed, A. Idries, N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "UAVs for smart cities: Opportunities and challenges," *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 267–273, may 2014.

[5] T. Castelli, H. Konik, and E. Dinet, "Moving object detection for unconstrained low-altitude aerial videos , a pose-independant detector based on Artificial Flow," *International Symposium on Image and Signal Processing and Analysis*, no. Ispa, pp. 44–49, 2015.

[6] F. Anon, V. Navarathinarasah, M. Hoang, and C. H. Lung, "Building a framework for internet of things and cloud computing," in *Internet of Things (iThings), 2014 IEEE International Conference on, and Green Computing and Communications (GreenCom), IEEE and Cyber, Physical and Social Computing(CPSCom), IEEE*, Sept 2014, pp. 132–139.

[7] S. Matuska, R. Hudec, and M. Benco, "The comparison of cpu time consumption for image processing algorithm in matlab and opencv," in *ELEKTRO, 2012*, May 2012, pp. 75–78.

[8] I. R. Ismaeil, A. Docef, F. Kossentini, and R. Ward, "Motion estimation using long-term motion vector prediction," in *Data Compression Conference, 1999. Proceedings. DCC '99*, Mar 1999, pp. 531–.

[9] J. L. Maryak, J. C. Spall, and B. D. Heydon, "Use of the kalman filter for inference in state-space models with unknown noise distributions," *IEEE Transactions on Automatic Control*, vol. 49, no. 1, pp. 87–90, Jan 2004.

[10] G. R. K. S. Subrahmanyam, A. N. Rajagopalan, and R. Aravind, "Unscented kalman filter for image estimation in film-grain noise," in *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, vol. 4, Sept 2007, pp. IV – 17–IV – 20.

[11] O. Javed, K. Shafique, and M. Shah, "A hierarchical approach to robust background subtraction using color and gradient information," *Workshop on Motion and Video Computing, 2002. Proceedings.*, 2002.

[12] K. Choi and J. Yun, "Robust and Fast Moving Object Detection in a non-stationary camera via foreground probability based sampling," *IEEE ICIP*, pp. 4897–4901, 2015.

[13] D. Hulens, J. Verbeke, and T. Goedem, "How to Choose the Best Embedded Processing Platform for on-Board UAV Image Processing ?" *Visigrapp*, pp. 377–386, 2015.

[14] F. Wang, J. Cui, S. K. Phang, B. M. Chen, and T. H. Lee, "A mono-camera and scanning laser range finder based UAV indoor navigation system," in *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, may 2013, pp. 694–701.

[15] Y. Ren, C. S. Chua, and Y. K. Ho, "Statistical background modeling for non-stationary camera," *Pattern Recognition Letters*, vol. 24, no. 1-3, pp. 183–196, 2003.

[16] G. R. Rodríguez-Canosa, S. Thomas, J. del Cerro, A. Barrientos, and B. MacDonald, "A real-time method to detect and track moving objects (DATMO) from unmanned aerial vehicles (UAVs) using a single camera," *Remote Sensing*, vol. 4, no. 4, pp. 1090–1111, 2012.

[17] F. Kendoul, I. Fantoni, and K. Nonami, "Optic flow-based vision system for autonomous 3d localization and control of small aerial vehicles," *Robotics and Autonomous Systems*, vol. 57, no. 67, pp. 591 – 602, 2009.

[18] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, pp. 359–392, 1998.

[19] S. Birchfield and C. Tomasi, "A pixel dissimilarity measure that is insensitive to image sampling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 4, pp. 401–406, Apr 1998.

[20] "MAVProxy: A UAV ground station software package for MAVLink based systems." [Online]. Available: http://dronecode.github.io/MAVProxy/html/index.html