Passive localization and detection of quadcopter UAVs by using Dynamic Vision Sensor

S. Hoseini[1], G. Orchard[2], A. Yousefzadeh[1], B. Deverakonda[2], T. Serrano-Gotarredona[1] and B. Linares-Barranco[1]

[1]Instituto de Microelectrónica de Sevilla (CSIC and Univ. de Sevilla), Sevilla, Spain {sahar, bernabe}@imse-cnm.csic.es
[2] Singapore Institute for Neurotechnology (SINAPSE) at National University of Singapore garrickorchard@nus.edu.sg

***Abstract*— We present a new passive and low power localization method for quadcopter UAVs (Unmanned aerial vehicles) by using dynamic vision sensors. This method works by detecting the speed of rotation of propellers that is normally higher than the speed of movement of other objects in the background. Dynamic vision sensors are fast and power efficient. We have presented the algorithm along with the results of implementation.**

Keywords*— *Dynamic Vision Sensors (DVS); Quadcopter UAV; Localization

I. INTRODUCTION

Frame-less Dynamic Vision Sensors (DVSs) are becoming more popular in vision processing systems due to their low power consumption and fast response time [1]. In conventional artificial vision systems, cameras capture frames at a given frame rate. In these systems, independent of the changes in the frames, the processing is repeated for every single frame. Therefore, for detecting fast moving objects, a high frame rate is required, resulting in a need for power hungry processing. However, in biological vision systems, the procedure is quite different. Our eyes contain many neurons that are sensitive to light and each neuron will generate a spike as soon as it detects a change in light.

There are different types of DVS with different resolutions and special features (for example [2] [3] [4]) but all of them share a common principle with biological retina. Each pixel independently sends events (or spikes) out when the change in light intensity exceeds a predefined threshold and it only consumes power in these moments (except for a low standby power in the absence of events). Output events of DVS are encoded in Address Event Representation (AER) format [5] that contains pixel address and a polarity bit. Polarity bit indicates the direction of change in light intensity. Fig.1 shows a propeller that rotates in front of DVS and the corresponding reconstructed DVS output in jAER software [6]. Black dots in reconstruction show decrease in light intensity and white dots show increase in light intensity. No event is be generated for fixed objects in background because of no change in light intensity.

Even though DVS is very low power, its response time is in order of microseconds and it can easily capture movements which only very fast frame-based cameras (>1000fps) can capture. In addition, DVS has very high dynamic range (120dB for [4], 130dB for [2] and 143dB for [3] in comparison to 50dB for typical frame-based cameras).

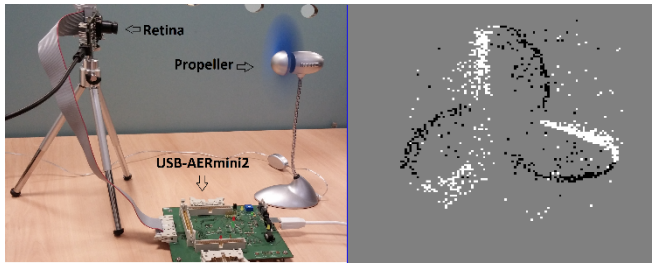


Fig.1. Rotating propeller in front of DVS [4] *(left)*. USB-AERmini2 board [5] puts time stamp on events from DVS and sends them to computer through USB. Output is reconstructed with jAER software (*right*) [6]. It contains 864 events within 624us (*1.3M events per second*).

Quadcopter applications are increasing and at the same time engineers are developing autonomous controllers to make them easier to use. Outputs of quadcopter localization systems can be used in these controllers. Quadcopter localization systems can also be used in security systems to detect quadcopters. Quadcopters can be as small as a bird and it is hard to detect them by RADAR systems. Quadcopters can be detected by using an intelligent system to distinguish them with birds based on motion estimation and other elusive aspects but with a limited accuracy. Due to use of electrical power (versus burning fuel) quadcopters do not produce a lot of heat (to be detected by thermal methods). Quadcopters can be completely autonomous and passive without producing any radio signals. These features make it hard for conventional systems to detect them. We propose to detect the high rotation speed of the propellers to make a robust system for detecting quadcopters. This method can be used alone or along with other methods to increase accuracy in security systems. We have introduced a fully event driven algorithm that is suitable for implementation in an embedded hardware.

Censi et al [7] introduced a low-latency localization method by using active LED markers. This method needs four flashing LEDs in a quadcopter. By detecting frequency of LEDs, they localized and tracked the quadcopter. We tried to remove the needs of using active LED markers and trying to recognize quadcopters by detecting frequency of rotating propellers. Because rotating propeller is harder to detect than

a flashing LED, we have designed the algorithm to be more robust against noise and proper for our application. The fact that our method is completely passive makes it adequate for detecting quadcopters in security applications. In the next sections we have explained the algorithm, experimental results and finally a brief conclusion.

II. DETECTION ALGORITHM

In this section we have explained the algorithm to detect frequency of a moving objects by using DVS. Because propellers normally move faster than other objects and their movement repeats periodically, our algorithm can distinguish it from other kind of movements. In this work we assumed that the DVS moves slowly over a fixed platform.

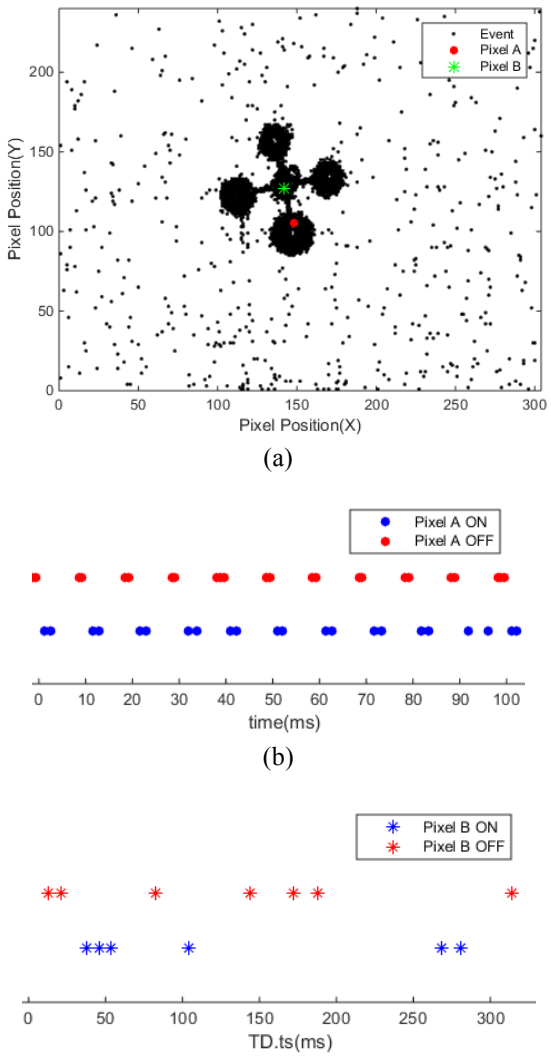


Fig.2. Output events of DVS when looking at a quadcopter for 100ms (a), 'ON' and 'OFF' events versus time for two pixels, one is looking to propeller (*Pixel A*) (b) and another one is looking to the frame of quadcopter (*Pixel B*) (c)

Fig.2 (a) shows the output events from all the pixels of DVS. We have selected two pixels to show the sequence of the events generated in time. One pixel is looking at the propeller (Pixel A) and another one is looking at the frame of quadcopter (Pixel B). As it can be seen in Fig.2 (b) and (c), Pixel A generates a good periodical series of 'ON' events (events with polarity==1) and 'OFF' events (events with polarity==0) while events from Pixel B do not have this feature. Detection of this feature is the principle of our frequency detection algorithm.

Events from DVS contain pixel address and polarity (p). Each event is processed based on its event address. Fig.3 is a simplified flowchart of the proposed algorithm to detect frequency of rotating object for only one selected pixel of DVS. Each pixel reports frequency of the object independent from others.

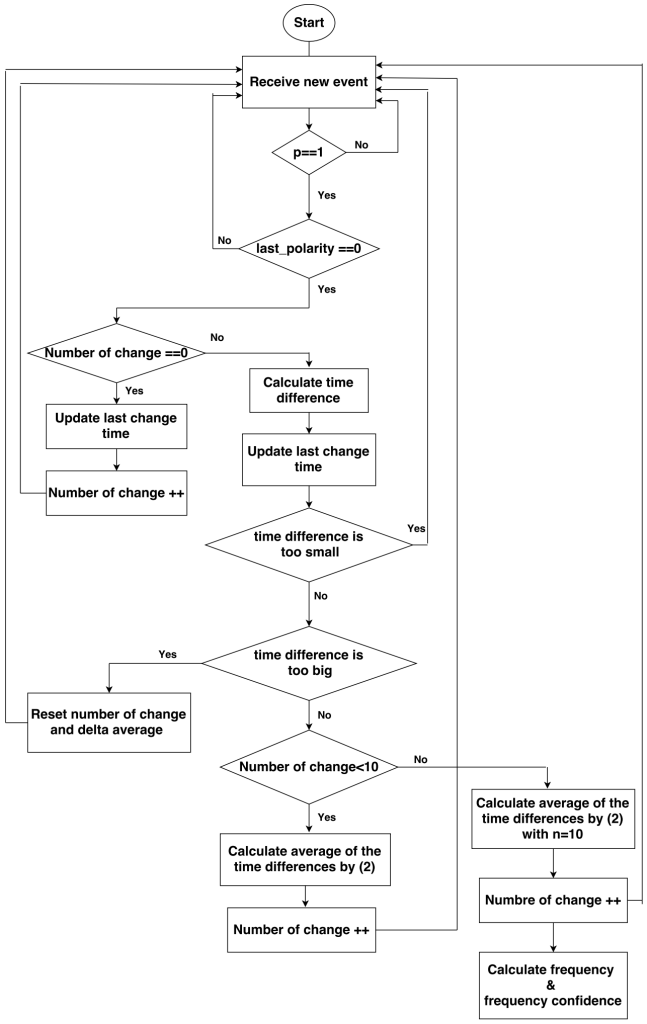


Fig.3. Flowchart of the proposed algorithm to calculate frequency of rotating object in front of a pixel of DVS

The algorithm is designed to calculate the time intervals between consecutive 'OFF' to 'ON' transitions as illustrated in Fig.4. At initialization, the system waits to receive an event with polarity equal to '1' (p==1). Then if the previous event from this pixel had polarity equal to '0' (last polarity==0) it will be considered as a change. Based on the number of change for each pixel, the algorithm will make different decisions to guarantee accuracy of results. If this is the first change for the pixel, t_{last} will be updated with t_{now}

and the number of change will be incremented by 1. Otherwise, the system calculates a time difference using (1).

$$\Delta t = t_{now} - t_{last}$$
$$t_{last} \leftarrow t_{now} \qquad (1)$$
$$n \leftarrow \begin{cases} n+1, & n < 10 \\ 10, & n \geq 10 \end{cases}$$

Where Δt is the time difference calculated using the current time, t_{now}, and the time of the previous event from this pixel, t_{last}. n is the number of changes detected at this pixel, or 10, whichever is smaller.

After updating t_{last} to t_{now}, it needs to check if the calculated time difference is reasonable. This condition acts like a frequency filter to remove noise. The reasonable range depends on the application and router frequency. In our experimental results, we adjust the system to accept frequencies in range of 5Hz to 200Hz (time difference in range of 5ms to 200ms). If time difference is less than 5ms, we assume it as a very fast change that can be from another source or internal noise of DVS. In this case, the system will ignore this change and does not consider it in the calculations. On the other hand, if the change is very slow, it means there is no rotating propeller in front of the selected pixel and in this case, the system will reset some internal states for this pixel to be ready for new stimuli.

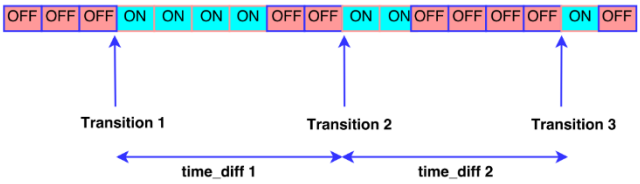


Fig.4. Definition of time difference between 'OFF' to 'ON' transition.

To suppress noise and jitter, the algorithm is designed in a way that it does not report frequency before receiving an adequate number of changes from the current stimuli. In this manner frequency will be reported only for pixels that sense reasonable repetitive movements. Here we adjusted it to receive at least 10 changes before reporting a frequency. Increasing this number will lead to more robust predictions but at the cost of increased latency.

Frequency will be reported by calculating average time of the past 10 changes. If the number of changes is less than 10, the average of the time differences ($\overline{\Delta t}$) will be calculated based on incremental averaging formula in (2). Incremental averaging allows us to calculate average in an event driven manner without storing previous events.

$$\overline{\Delta t} \leftarrow \frac{(n-1)\overline{\Delta t} + \Delta t}{n} \qquad (2)$$

If the number of changes is more than 10, the system can start reporting frequency for the current stimuli. In this case $\overline{\Delta t}$ should be calculated from (2) with constant value of n (n=10). It means to calculate $\overline{\Delta t}$ we just consider 10 last changes rather than all of the previous events. Frequency is not constant and can change from one value to another. The ideal case to remove noise and not accumulating errors is using moving average and consider only recent events. Calculating exact moving average without knowing the previous events is not possible. We proposed an approximation to calculate moving average in an incremental manner by using (2) and set window size of moving average to a fixed value by setting n=10.

Results show good accuracy in real tests that will be discussed in the next section. After updating the $\overline{\Delta t}$, frequency will be calculated in (3).

$$f = \frac{1}{\overline{\Delta t}} \qquad (3)$$

In addition to frequency, the system will also report frequency confidence. This value presents confidence of the system about reported frequency and can be calculated with (4). If the current time difference is very different from the average of recent ones, frequency will be reported with less confidence.

$$\textit{Frequency Confidence} = 1 - \left| \frac{\Delta t - \overline{\Delta t}}{\overline{\Delta t}} \right| \qquad (4)$$

Proposed algorithm is pure event driven. When there is no input event, no processing is needed. To save memory we eliminated the need to store previous events and it is only needed to store four values for each pixel in the hardware:

1- Polarity of previous event (p_{last})
2- Time of previous polarity change (t_{last})
3- Total number of polarity change (n)
4- Average of time between changes ($\overline{\Delta t}$)

These are the state of each pixels and their initial value is zero. In the next section we explain the implementation results.

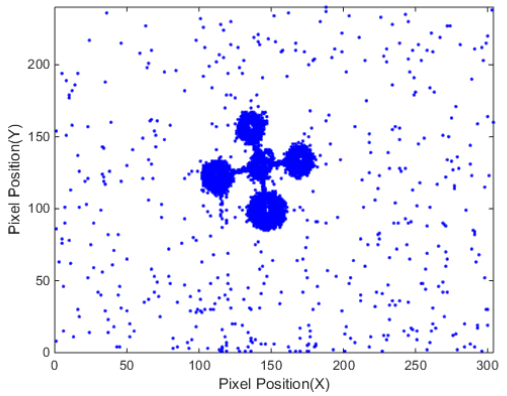
III. Implementation Results

To test the algorithm, we used a 304×240 pixel DVS (ATIS) [3] and recorded its output events when it was looking at a flying middle size quadcopter as shown in Fig.5.

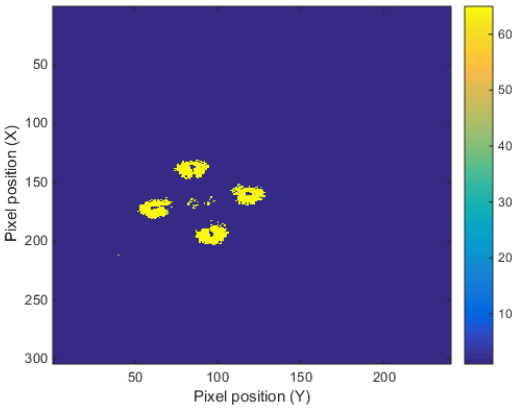
Fig.6 (a) shows a captured moment (200ms) of input events, Fig.6 (b) and (c) show one moment of output of the system (frequency of all the pixels vs. pixel position) in 2 and 3 dimensions respectively. As it can be seen, only the position of propellers in the quadcopter are highlighted and other positions do not have reported frequency. This results shows how robust is the system in a noisy environment.

Latency of recognition is low and reasonable. For the parameters that we chose in this work, minimum 10 changes are required for recognition. If the propeller's frequency is more than 50Hz, latency of recognition will be smaller than 200ms. To see the functionality of the system when the quadcopter is moving a real-time demo was prepared [8].

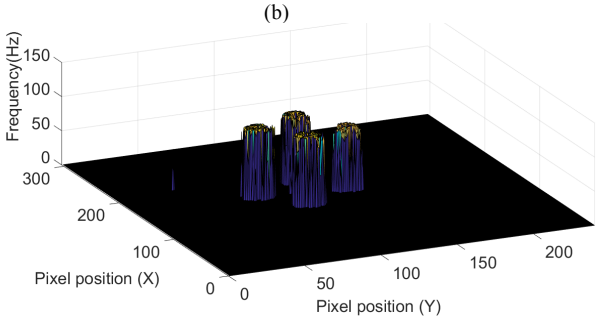
Fig.5. Quadcopter UAV that is used in this work to test the algorithm



(a)



(b)



(c)

Fig.6. Input events from a flying quadcopter (a), two-dimension output result (*color bar shows the approximate value of frequency*) (b), three-dimension output result(c)

Additional tests on the system have been done to verify the dynamics of the algorithm for moving from one frequency to another and testing frequency confidence. For this test, we used recorded events from a flashing LED with a PWM controller to change its frequency. We used this recording because controlling propeller rotation speed is not easy in a flying quadcopter.

Fig.7 shows results of the algorithm for only one pixel in front of a flashing LED. As expected sometimes frequency confidence is less at higher frequencies. This is because the DVS needs a small amount of time for integration and event generation. In high frequency change in light intensity, a DVS starts losing events as it can be seen in Fig.8. By fine tuning the DVS parameters (for example threshold voltage) it can be adjusted for work in at higher frequency but it may generate more noisy events. Also when frequency changes from one value to another, the frequency confidence will decrease for a short time.

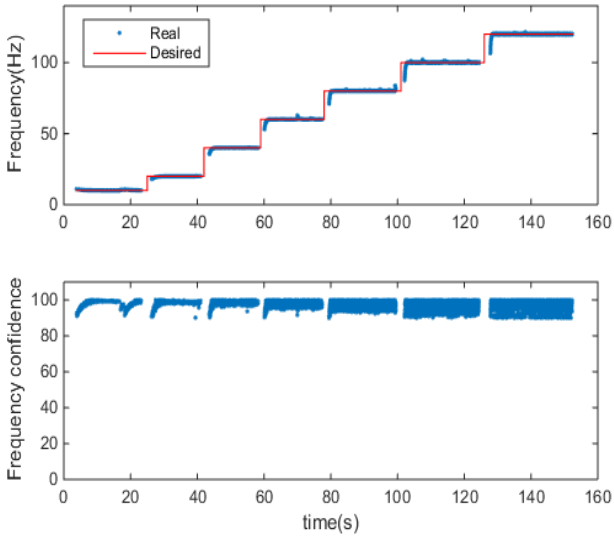


Fig.7. Result of frequency detection of one pixel of blinking LED

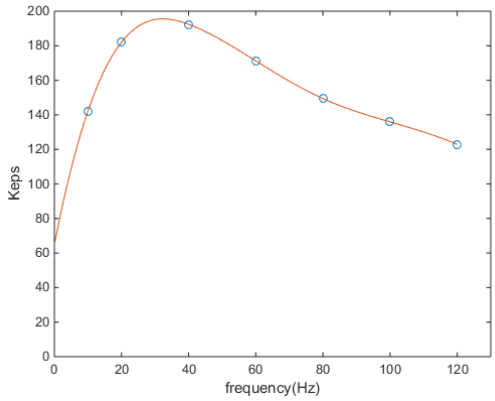


Fig.8. Average number of events per second from flashing LED which is captured by DVS.

Current implementation of the algorithm in MatLab can process around 64Keps (thousands of events per second) in an ordinary laptop while real-time event rate for our flying quadcopter test is around 124Keps. Because in this algorithm

events from different pixels can be processed in parallel, we are planning to implement this algorithm in an embedded multi-core ARM processor to achieve real-time processing.

IV. Conclusion

In this work, we introduced a passive method to localize and detect quadcopter UAVs by using a Dynamic Vision Sensor. This algorithm can be efficiently implemented in an embedded hardware to detect UAVs with rotation wings and distinguish them than other flying animals like birds. Also it can be used in noisy outdoor environment to control UAVs.

V. Acknowledgements

This work has been supported by Singapore Institute for Neurotechnology (SINAPSE).

VI. Refrences

[1] S. Soman, Jayadeva, M. Suri, "Recent trends in neuromorphic engineering," *Big Data* Analytics*,* Dec 2016.

[2] R. Berner, M. Yang, S. Chii-Liu, T. Delbruck, C. Brandli, "A 240 × 180 130 dB 3 µs Latency Global Shutter Spatiotemporal Vision Sensor,," *IEEE Journal of Solid-State Circuits,* vol. 49, no. 10, pp. 2333-2341, Oct 2014.

[3] D. Maolin, R. Wohlgenannt, C. Posch, "A QVGA 143dB dynamic range asynchronous address-event PWM dynamic image sensor with lossless pixel-level video compression," in *IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, 2010.

[4] T. Serrano-Gotarredona and B. Linares-Barranco, "A 128x128 1.5% Contrast Sensitivity 0.9% FPN 3us Latency 4mW Asynchronous Frame-Free Dynamic Vision Sensor Using Transimpedance Amplifiers," *IEEE J. Solid-State Circuits,* vol. 48, no. 3, March 2013.

[5] T. Serranno-Gotarredona, M.Oster, P.Lichtsteiner, R.Paz-Vicente, F.Gomez-Rodriquez, B. Linares-Barranco, et al "CAVIAR: A 45k Neuron, 5M Synapse, 12G Connects/s AER Hardware Sensory–Processing– Learning–Actuating System for High-Speed Visual Object Recognition and Tracking," *IEEE Transactions on Neural Networks,* Sept. 2009.

[6] F.Corradi, S.Bamford, L.Longinotti , T.Delbruck"jAER," [Online]. Available: https://sourceforge.net/projects/jaer/.

[7] A. Censi, J. Strubel, C. Brandli, T. Delbruck, D.Scaramuzza, "Low-latency localization by Active LED Markers tracking using a Dynamic Vision Sensor," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, 2013.

[8] S. Hoseini, Sep 2016. [Online]. Available: https://youtu.be/RFcjUBN3pw0.