Live Demonstration: A Real-time Event-based Fast Corner Detection Demo based on FPGA

Min Liu, Wei-Tse Kao, Tobi Delbruck
Institution of Neuroinformatics, University of Zurich and ETH Zurich
{minliu@ini.uzh.ch}

Abstract

*Corner detection is widely used as a pre-processing step for many computer vision (**CV**) problems. It is well studied in the conventional CV community and many popular methods are still used nowadays such as Harris, FAST, and SIFT. For event cameras like Dynamic Vision Sensors (**DVS**), similar approaches also have been proposed in recent years. Two of them are event-based Harris (**eHARRIS**) and event-based FAST (**eFAST**). This demo presents our recent work in which we implement eFAST on MiniZed FPGA. The power consumption of the whole system is less than 4W and the hardware eFAST consumes about 0.9W. This demo processes at least 5M events per second, and achieves a power-speed improvement factor product of more than 30X compared with CPU implementation of eFAST. This embedded component could be suitable for integration to applications such as drones and autonomous cars that produce high event rates.*

1. Brief Introduction and Motivation

Corner detection or keypoint extraction is a basic but important topic in computer vision. It extracts the most important information from the image data to avoid processing all pixels and thus decreases the latency and quality of subsequent processing. It underlies many optical flow, visual odoometry, and SLAM methods. There are several corner detection methods adapted for event sensors. Two of them are event-based Fast (**eHARRIS**) [3] and event-based Harris (**eFAST**) [4]. Compared with eHARRIS, eFAST requires less expensive computation since it does not require derivative calculations. Most of the operations from eFAST are comparison, sorting and accumulating. Based on this, we choose to target eFAST for logic circuit implementation.

The motivation of this work is that we want to design a general pre-processing real-time hardware logic module (**IP**)[1] for event-based algorithms to further reduce the latency and CPU load on host processor. DVS event cameras are known for their sparse output, high temporal resolution and high dynamic range. Each single event contains little information (although on the average more than a conventional pixel sample), so it is not necessary to process every event. Some key events can be detected so as to only compute subsequent information around these events. We first conceived this aim as a sub project of our previous event-based optical flow work (**ABMOF**)[2]. ABMOF implemented on FPGA currently calculates the optical flow for every event at about 1M events per second. For some fast motions such as panning the image across densely textured scenes, the rate goes above 10MHz. ABMOF originally proposed to simply downsample (by skipping) events that flow is computed on. eFAST based on FPGA is intended in this context for more informative selection of relevant events on which to compute flow.

2. Experimental Result

To implement the eFAST IP block, we used high level synthesis (Vivado HLS[5]) rather than the lower-level (but more efficient) System-Verilog from Xilinx. HLS is a very convenient tool for software engineers who wants to accelerate their work on FPGA. The grammar of HLS is similar to C/C++ but the designer must be familiar with the capabilities of hardware parallelism and memory on the FPGA for effective results.

The setup of our demo is shown in Figure 1. It mainly consists of five parts: DAVIS240C[1], MiniZed[2] hosting entry level $90 Xilinx Zynq 7z007s SoC FPGA, power meter, wireless router and the laptop. The DAVIS is connected to the MiniZed via its one USB interface. eFAST is implemented on the MiniZed logic. A USB serial port interfaces UART to computer for logic debug and the last USB provides power. A minimal PetaLinux linux distribution is installed on the MiniZed processor to use libcaer[3] to fetch the DAVIS data and to transmit the results by UDP over WiFi.

[1] IP is used in logic design community for "intellectual property" block.

[2] http://zedboard.org/product/minized

[3] https://github.com/inivation/libcaer

The MiniZed is connected to a power meter.

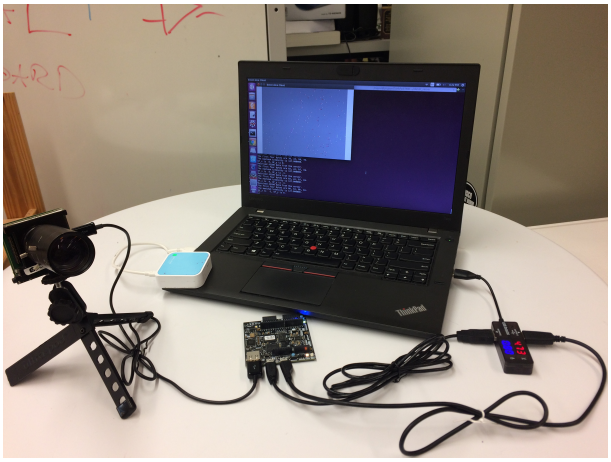

Figure 1. The demo setup

Detected keypoints cannot be rendered on MiniZed because it lacks any display interface. Instead, eFAST output is sent to a laptop by WiFi connection using a dedicated router for the laptop and MiniZed to reduce the transfer latency. The laptop hosts a C++ server for display of the keypoint detector and DVS data. The laptop is only used for rendering results and providing power.

We measure the process time and power consumption on our demo and compared it with eFAST running on a Ubuntu 16.04 PC with a quad-core 2.3GHz CPU (i5-8259U). eFAST on PC used the implementation of [3][4], which runs inside ROS. It can output the average processing time per event after every packet is processed.

For the power consumption measurement, we use a power meter for MiniZed and the software tool *powertop*[5] for laptop. We first measure the static power without running eFAST on both of them. After starting the algorithm, the differential value of the power meter and powertop is the power consumption resulting from eFAST.

Table 1. comparison between eFAST on FPGA and PC

platform	FPGA	PC	Comparison
processing time/us	0.1-0.2	0.6-30	>**6x** faster
eFAST power/W	0.9	5	**5x** more efficient
total power/W	3.7	17.5	**5x** more efficient

Results are shown in Table 1. There are three metrics: processing time, eFAST power, and total system power. *eFAST power* only takes into account the eFAST. *Total power* counts all the parts (DAVIS and MiniZed, or laptop). eFAST running on this small platform is more than $6\times$ faster than running on PC and requires $5\times$ less power consumption. Therefore, the FGPA implementation achieves a $\text{power} \times \text{speed}$ improvement product that is more than $30\times$ higher in FPGA than PC. With 100MHz clock, the processing time for each event is either 100ns or 200ns (depending on local context), which means that this system handles up to 10M events per second which is sufficient for most real applications. The eFAST utilized 25% Look up tables (LUTs), 11% Flip-Flops (FFs) and 57% Block RAMs on the MiniZed.

3. Conclusion

We presented a real-time eFAST corner detection demo for event-based camera. This demo processes up to 10MHz event rate. The eFAST IP will serve as a useful preprocessing module of many CV problems requiring real-time performance such as Optical Flow, Visual Odometry and SLAM.

We plan to soon release this IP for free non-commercial exploitation.

4. Acknowledgement

This work was supported by the Swiss National Centre of Competence in Research (NCCR) Robotics.

References

[1] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240× 180 130 db 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014.

[2] Min Liu and Tobi Delbruck. Adaptive time-slice block-matching optical flow algorithm for dynamic vision sensors. In *British Machine Vis. Conf.(BMVC)*, 2018.

[3] Elias Mueggler, Chiara Bartolozzi, and Davide Scaramuzza. Fast event-based corner detection. In *British Machine Vis. Conf.(BMVC)*, volume 1, 2017.

[4] Valentina Vasco, Arren Glover, and Chiara Bartolozzi. Fast event-based Harris corner detection exploiting the advantages of event-driven cameras. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4144–4149. IEEE, 2016.

[5] Vivado-HLS Xilinx. Vivado design suite user guide-high-level synthesis, 2018.

[4] https://github.com/uzh-rpg/rpg_corner_events

[5] https://github.com/fenrus75/powertop