Advances in the development of a low-cost video meteor station

Dario Zubović[1], Denis Vida[2], Peter Gural[3] and Damir Šegon[4]

[1] **Croatian Meteor Network**
dario@zubovic.email

[2] **Astronomical Society "Anonymus", B. Radića 34, 31550 Valpovo, Croatia**
Faculty of Electrical Engineering, University of Osijek, Kneza Trpimira 2B, 31000 Osijek, Croatia
denis.vida@gmail.com

[3] **International Meteor Organization**
peter.s.gural@leidos.com

[4] **Astronomical Society Istra Pula, Park Monte Zaro 2, 52100 Pula, Croatia**
damir.segon@pu.htnet.hr

Recent advances in the field of single board computers, have enabled the development of a low-cost video meteor station with real-time processing capabilities. In this paper, an overview of different capture and computing hardware is given. Furthermore, we present the current state of new open-source software for video meteor capture, multi-frame compression and real-time detection. The software is compatible with the existing Croatian Meteor Network processing package.

1 Introduction

Studying meteors with the video systems within the Croatian Meteor Network began in 2007 (Andreić, 2009). Since then, the emphasis has been on building inexpensive stations in order to deploy as many of them as possible. US$45 CCTV cameras with sensitivity comparable to the several times more expensive cameras such as the Watec 902H2 have already been presented (Samuels, 2014). But the most expensive part remained the computer performing the meteor video acquisition and processing. The proposed system design herein integrates a full setup of CCTV video camera, USB frame grabber, and single board computer (SBC) comprising a small form factor and priced at under US$150. The entire system fits within a standard CCTV camera housing with only external power and Ethernet connectivity required, which lends itself to high mobility and simple deployment. First prototype is shown in *Figure 5*.

2 Hardware platform

Single-board computers

The price and desire for real-time processing drove the selection of the SBC that was used in the final design. The Raspberry-Pi 1B[1] and Banana-Pi 1[2] did not meet those requirements, but provided a good testing platform before faster hardware appeared on the market in late 2014. The Raspberry-Pi 2[3] with 4 CPU cores running at 900 MHz made parallelized processing possible that allowed capture, compression and detection to run on separate threads. In order to utilize this hardware, an initial end-to-end software development task was completed, with the testing of better algorithmic solutions an ongoing effort.

Table 1 – The list of tested single-board computers.

Name	CPU	RAM	Price
Raspberry Pi 1 B	700 MHz	512 MB	$35
Banana Pi 1	1 GHz	1 GB	$40
Raspberry Pi 2	4 cores @ 900 MHz[4]	1 GB	$35
Odroid C1	4 cores @ 1.5 GHz	1 GB	$35
Orange Pi 2 Mini	4 cores @ 1.6 GHz	1 GiB	$25

Two additional SBCs were tested by timing the compression algorithm which was executed 20 times and the results averaged. The Odroid C1[5] running at 1.5GHz seemed promising, but performed 11% slower than the Raspberry-Pi 2. In the worst-case scenario (256 sequential white images), the Odroid C1 performed 2% better than the Raspberry-Pi 2. Those results probably have to do with their different architectures. Also, problems appeared when the RAM was more than 50% utilized. Since the RAM on the Odroid C1 is split across two chips, the memory management schema is likely to be blamed.

[1] https://www.raspberrypi.org/products/model-b/
[2] http://www.lemaker.org/article-42-1.html
[3] https://www.raspberrypi.org/products/raspberry-pi-2-model-b/
[4] Overclocked to 1GHz.
[5] http://www.hardkernel.com/main/products/prdt_info.php?g_code=G141578608433

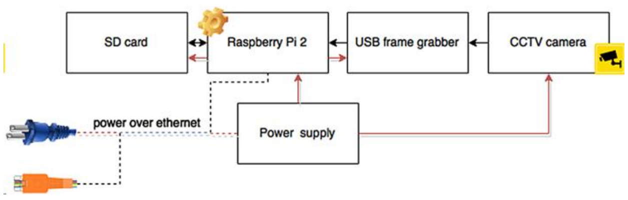

Figure 1 – System design.

Orange-Pi Mini 2[6], with a slightly lower price than Raspberry-Pi 2, was the next device tested. While running the compression algorithm, its CPU would overheat as soon as the single core reached 100% usage. Both the Orange-Pi Mini and the Odroid C1 demonstrate that allegedly more powerful hardware should not be the only selection criteria, but their hardware/software support should be considered as well. In the case of open-source software, it's mostly the community of SBC users that is making a difference. Thus, we decided to use the Raspberry-Pi 2 and tap into the associated community of developers whom own more than 5000000 devices[7].

Frame grabbers

Since low-cost analog frame-rate CCTV cameras are used, an accompanying digitizer is required to convert the video into digitized frames. The EasyCap frame grabbers, which can often be bought for about $10, provide an excellent solution but there are 4 completely different clones all sold under the name “EasyCap”. To make things worse, only some work under Linux[8].

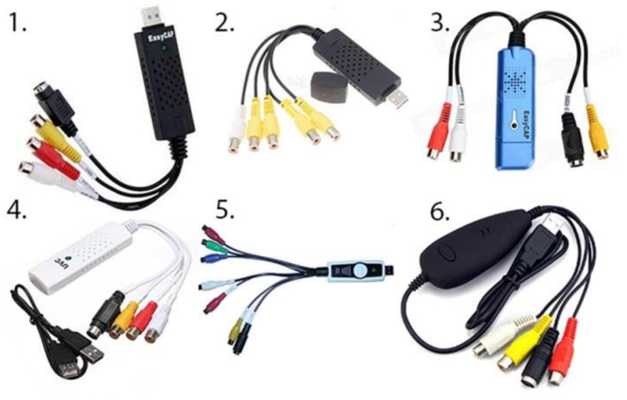

Figure 2 – Different designs of USB frame grabbers.

Six devices have been identified that could potentially be used and are shown in ***T****able 2*. Out of the four that were actually tested under Linux, three were found to work with the Raspberry-Pi 2, although only the one containing the chipset UTV007 has been found to be fully compatible with the software described below. There are sellers on AliExpress that have “UTV007” declared on their product page. The SMI-2021 chipset version required the extraction of the property binary-blob in order to get V4L2 drivers working, and the kernel had to be compiled with additional modules. Arkmicro worked out of the box, but is limited to 640x480 pixel digitization and the output video stream is compressed with MJPEG, instead of being raw bytes stored in computer memory. Those devices that can be easily identified by their chipsets before purchase are shown in *Figure 2*. *Figure 1* displays system design schematic.

Table 2 – Low-cost frame grabbers

Brand name and chipset	Working on Raspberry Pi 2	USB ID	Identification (see *Figure 2*.)	Price
EasyCap - STK1160	No	05e1:0408	1.	$10
EasyCap - UTV007	Yes	1b71:3002	1.	$8
EasyCap - SMI-2021	Limited	1c88:0007	1. or 2. or 3.	$9
EasyCap - EM2860	Untested	eb1a:2861	1. or 6.	$12
Various names - EM28284	Untested	eb1a:8285	5.	$16
Various names, often “EasyCap UVC” - Arkmicro	Limited	18ec:5850	4.	$6

3 Software solution

Open-source software[9] was custom written for the purpose of utilizing SBCs in video meteor projects. While time critical pieces of code are written in C++, most of the codebase is written in Python. Due to the nature of Python, software can be easily exported to other computing platforms with minimal modifications. The goal was that the software include at least the same features found on current CMN stations and generate output products in the same format. Faint meteor detection is still under development and will be discussed in a future paper.

Capture and compression

The current processing pipeline is shown in *Figure 3*. Interfacing with the V4L2 drivers of the capture device is achieved through Python bindings for OpenCV. A 256 sequential frame block is stored entirely in memory before being passed to the compression algorithm which runs on a separate core. During compression processing, the capture continues storing the next 256 frame block without interruption (asynchronous processing). The compression method chosen was initially developed by Mark Vornhusen for the SkyPatrol software application and then later refined for the CAMS project (Gural, 2010) as shown in *Figure 4*. The compressed image products (max pixel, frame number of max, mean, and standard deviation) are kept in memory for further processing and also stored to a hard drive in the same binary format used for CAMS, thus maintaining full compatibility with existing tools such as CMN_BinViewer (Vida, 2014).

[6] http://www.orangepi.org/orangepimini2/
[7] https://www.raspberrypi.org/blog/five-million-sold/
[8] http://linuxtv.org/wiki/index.php/Easycap
[9] https://github.com/CroatianMeteorNetwork/RMS

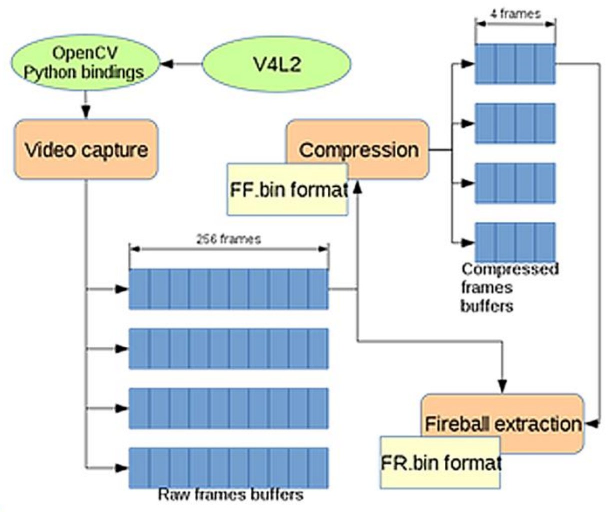

Figure 3 – Processing pipeline.

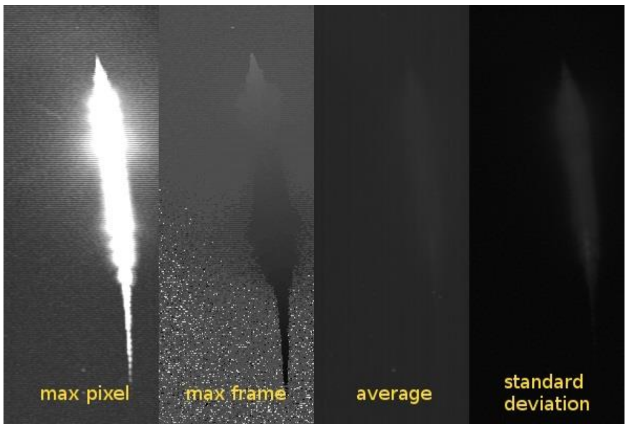

Figure 4 – Compressed image products from a 256 frame sequence.

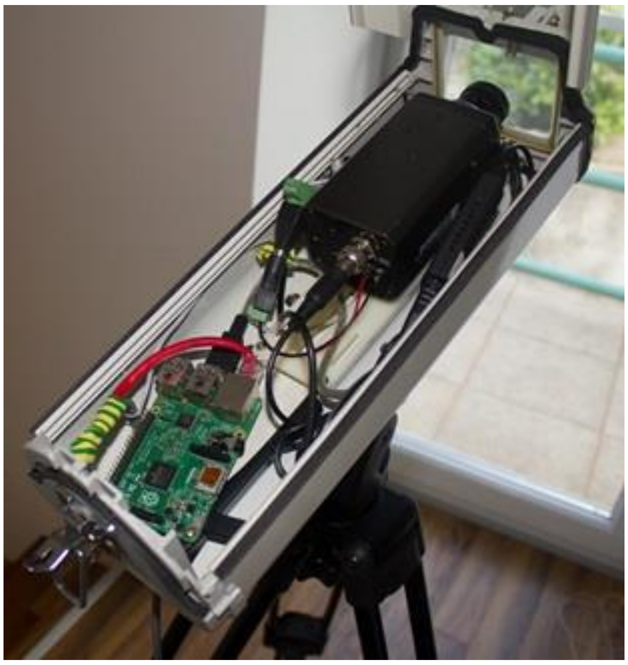

Figure 5 – Assembled prototype with all components contained within a standard video camera housing.

Raw video extraction

The compressed image is both thresholded and subsampled to 16x16 blocks in order to speed up calculations as shown in *Figure 6*. The subsampling was inspired by the AIM-IT project (Gural, 2004). Time information (frame number) is stored in the compressed image set and represents a third dimension in addition to the two spatial dimensions. Thus, line finding in 3D can be performed. A fast algorithm was developed for 3D line finding based upon the RANSAC algorithm (Fischler, 1981). That algorithm, although written in Python, is executing fast enough to enable real-time fireball detection on compressed images. It's highly tolerant to noise, but can produce false-positives. When the line finding algorithm returns a positive result, the raw frames which still exist in memory, are cropped around the propagating detection and saved in a custom format. The size of cropped region is calculated dynamically based upon the meteor's brightness and thus not fixed in dimension. The fireball detection and extraction process runs on separate CPU core relative to capture or compression. It enables storing the raw image chip outs for important fireball events, without flooding storage with full-frame raw video.

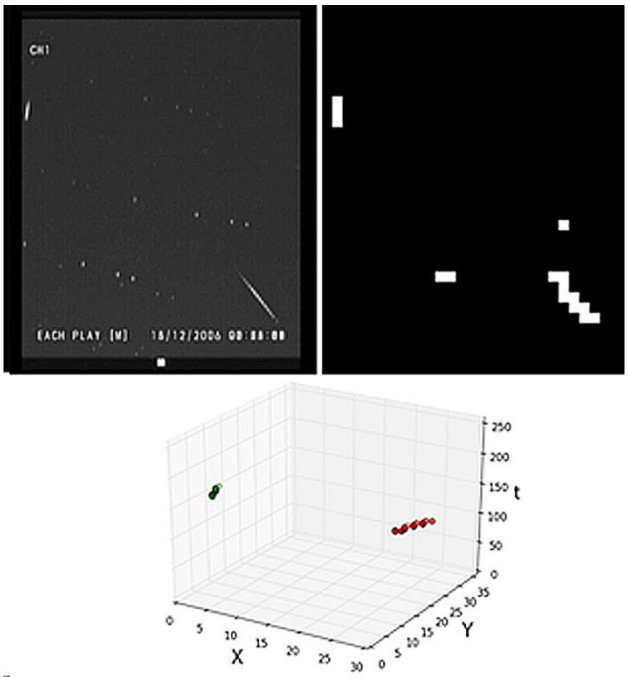

Figure 6 – Top-left: max pixel for test data of two meteors. Top-right: thresholded and subsampled image. Bottom: 3D line finding algorithm results.

4 Conclusion

The primary goal of building a prototype all-in-one meteor camera system was successfully achieved with the cost for all components falling under the target price of US$150. The hardware and software requirements have been met although still under active development. The authors hope that the described system design and software will be tested and further refined by other video meteor observers. With current trends in technology, like faster multi-core SBCs and the use of open-source software development, video meteor astronomy can move forward with very compact systems at far lower cost than previously experienced.

References

Andreić Ž., Šegon D. and Korlević K. (2010). “The second year of Croatian Meteor Network”. In Andreić Ž. and Kac J., editors, *Proceedings of the International Meteor Conference*, Poreč, Croatia, 24-27 September 2009. IMO, pages 26–30.

Gural P. S., Jenniskens P. M., and Varros G. (2004). “Results from the AIM-IT meteor tracking system”. *Earth Moon Planets*, **95**, 541–552.

Gural P. S. (2011). “The California All-sky Meteor Surveillance (CAMS) System”. In, Asher D. J., Christou A. A., Atreya P. and Barentsen G., editors, *Proceedings of the International Meteor Conference*, Armagh, Northern Ireland, 16–19 September 2010. IMO, pages 28–31.

Fischler M. and Bolles R. (1981). “Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography”. *Commun. Assoc. Comp. Mach.*, **24**, 381–395.

Samuels D., Wray J., Gural P. S., and Jenniskens J. (2014). “Performance of New Low-cost 1/3" Security Cameras for Meteor Surveillance”. In Rault J.-L. and Roggemans P., editors, *Proceedings of the International Meteor Conference*, Giron, France, 18–21 September 2014. IMO, pages 66–73.

Vida D., Šegon D., Gural P. S., Martinović G. and Skokić I. (2014). “CMN_ADAPT and CMN_binViewer software”. In Rault J.-L. and Roggemans P., editors, *Proceedings of the International Meteor Conference*, Giron, France, 18–21 September 2014. IMO, pages 59–63.

The author, *Dario Zubović*, during his lecture (Photo by *Axel Haas*).