DOI 10.5162/ETTC2024/A9.2

Introduction of Smart Hi-Speed Airborne Camera with Embedded Real-Time Tracking Algorithm for Store Separation Flight Tests

Rodrigo C. de Paulo[1,2], Nelson P. O. Leite[1], Lucas B. R. Sousa[1], Denis S. Loubach[2]
[1] Instituto de Pesquisas e Ensaios em Voo (IPEV), Pça Mal. Eduardo Gomes nº 50, Vila das Acácias, São José dos Campos, SP, 12.228-901 - Brazil,
[2] Instituto Tecnológico de Aeronáutica (ITA), Pça Mal. Eduardo Gomes nº 50, Vila das Acácias, São José dos Campos, SP, 12.228-900 - Brazil
rodrigorcp@fab.mil.br, nelsonnpol@fab.mil.br, lucaslbrs@fab.mil.br, dloubach@ita.br

Abstract:
For efficiency and flight safety improvement in Store Separation Flight Testing (SSFT) IPEV´s R&D efforts over the past 10 years were focused on the development and validation of an Airborne Real-Time Optical Tracking System (SisTrO), to provide the ejected store Time-Space Position Information (TSPI) in real-time and expressed in 6 Degrees-of-Freedom (6DoF) parameters (i.e., 3-D Linear and Angular Displacement). The current SisTrO architecture employs Hi-Speed airborne cameras connected to a tracking processor through the CoaXPress (CXP-12) interface. The recent introduction of airborne hi-speed cameras where the users can integrate their own application into the camera FPGA, opens new frontiers where the tracking algorithm is embedded into the camera processing unit, making possible the development of "Hi-Speed airborne smart cameras" to be used in flight testing, such as SSFT campaigns to provide a real-time efficient solution in a smaller form factor. Furthermore, using Perspective-n-Point techniques opens new frontiers for developing an advanced algorithm to compute the 3D 6DoF trajectory parameters with sufficient accuracy from a single camera and therefore to improve SisTrO reliability and in a smaller form factor. Given this context, the present paper evaluates the implementation of different tracking algorithms in CPU/GPU used in SisTrO I/II with the new small-factor FPGA hardware architecture. The evaluation parameters are real-time performance (i.e., execution time), object tracking effectiveness, and operational environment (flight tests). Results show that while the SisTrO CPU/GPU architecture has better computation performance, the FPGA can execute the tracking algorithms fulfilling the real-time constraints with reduced form factor and weight.

Keywords: Store Separation Flight Test, Photogrammetry, Object Tracking, Hardware Architecture.

Introduction

Store Separation Flight Tests (SSFT) are fundamental to guarantee a safe separation condition of the load over its entire operational envelope. Since there is a real chance of collision of the store with the aircraft due to unexpected or unknown aerodynamic coupling, and wing servo-aeroelasticity flexion in a given Store Separation Flight Test it should be verified if the store has some tendency to collide with the aircraft.

To perform this task the Instituto de Pesquisas e Ensaios em Voo (IPEV – "Flight Tests and Research Institute" (IPEV) has used high-speed camera image frames to evaluate and compute the trajectory of the store after launching.

In general, photogrammetry techniques [01] are often used to reconstruct the store trajectory in time-consuming post-mission operations, which may take hours or even days. To reduce flight hours and flight test campaigns, costs, reworking, and re-flights and to allow 2 or more launches in the same flight, IPEV has been developing and validating an Airborne Real-Time Optical Tracking System (SisTrO) to provide the ejected store Time-Space Position Information (TSPI) in real-time and expressed in 6 Degrees-of-Freedom (6DoF) parameters.

After two successful campaigns, with two different approaches, one as an improvement of the other, IPEV can now perform real-time object tracking with two different system architectures embedded in a photogrammetric POD. However,

with the evolution of hardware architecture and systems as well as sensors, nowadays it is possible to find COTS high-speed cameras that allow users to access the camera processing unit, in general, by giving access to its reconfigurable hardware (e.g., FPGA), which permits the users to integrate their own application on-site to run into the camera.

This work uses such advances in hardware architectures and sensors to develop a novel real-time object tracking system in small-factor (centimeters units) by employing the expertise and knowledge gathered in the previous campaigns by using a novel high-speed reconfigurable camera, which not only allows real-time processing but also reduces the equipment count and volume required to perform this task. In summary, this work presents the design of an object tracking system in FPGA and compares it with the previous developments that used a different hardware architecture.

SisTrO I

IPEV has developed and validated two versions of the Optical Tracking Systems (SisTrO) described as follows.

The initially pursued architecture was derived from an application presented in ETTC 2013 that used an iPhone 4S (Figure 01) to track and compute aircraft trajectory in the Air Data System calibration Flight test campaign by using the Tower-Fly-By method [2]. In such application, the deep integration between the video camera and processor resulted in satisfactory real-time performance where the time to compute the aircraft altitude and speed of a 720p image frame was 599ms±19ms @1σ, exceeding the camera connected to a processor traditional architecture by using an Ethernet connection.

Fig. 01 - Screen of iPhone4S Snapshot.

The first development was the SisTrO I, which was successfully tested in a flight test campaign in 2019 and performed object tracking in real time. The initial objective was to use an airborne hi-speed (i.e., up to 400 fps) hi-resolution (i.e., 1080p) camera with provisions for integrating a user application into the camera processing unit. However, at that time, finding a supplier with such a camera was not possible, mostly due to Intellectual Property (IP) issues. Therefore, the solution was to use a processing unit connected to a camera. However, the asynchronous Gigabit Ethernet connection introduces significant delays in hi-rate (i.e., 19.91Gb/s) data transfers of image frames, resulting in unsatisfactory performance for real-time applications.

The introduction of the CoaXPress (CXP-6) protocol [3] that supports up to 25 GB/s sustained (i.e., 100% duty cycle) data transfers solved the issue of transferring real-time live image frames from the high-speed camera to an image processor. After a long search, it was possible to build such processor with a CXP-6 interface fully compliant with MIL-STD-810G for airborne applications.

Thus, it was possible to define SisTrO I architecture, which was composed of (Figure 02):

1. Two high-speed cameras, each one connected to an image processor, which detects and tracks the store Reference Marks (RM); and
2. A Trajectory Processor, that computes the 3D trajectory from the detected 2D coordinates of the tracked RM; and
3. An IRIG-B GNSS time base and PtP v.2. grandmaster.

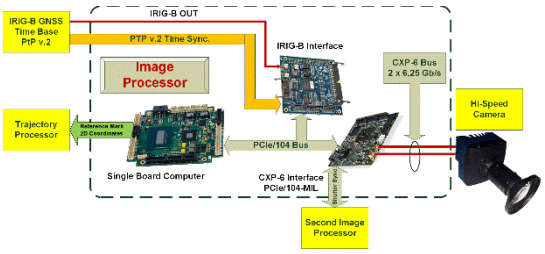

Fig. 02 – SisTrO I Architecture

SisTrO I uses the CAMSHIFT technique to perform the 2D tracking of RM. It is initialized with the positions of the RM in the first frame and searches in sequential frames the corners in a local window.

Although SisTrO I RM 2D measured tracking performance (i.e., 75,13%) provides enough satisfactory information for computing the 3D trajectory of the ejected store in all flight test points, it was observed that once the tracking

algorithm can't detect a given RM, such RM is permanently lost for all remaining image frames.

SisTrO II

SisTrO II development was aimed to improve SisTrO I tracking and real-time performance and also to encompass 3D solution redundancy. In this case, the image processor executes the 2D RM detection & tracking and the 6DoF store trajectory estimation by using a RANSAC Perspective-N-Point algorithm.

To improve 2D tracking the enhanced algorithm uses an ArUco Markers [4] for image segmentation (i.e., for area selection where each RM is most possible located) and neural network techniques. Such solution made it possible to reacquire a given RM tracking that was not identified in previous image frames due to image blur or low/excessive light and as a consequence improve algorithm 2D performance.

After image segmentation, the algorithm performs sub-pixel corner detection for RM tracking. A Convolutional neural network (CNN) was designed and trained to perform the marker identification obtaining successful results as detailed in [5], however, the flight test campaign was executed with the classical computer vision algorithm OpenCV cornersubpixel.

Corner Detectors

There are several ways to identify an object. In classical computer vision, it is common to use some features like edges, corners, and contours. More specifically, corners have proved to be a good feature to track as demonstrated in the Harris corner detector [6], the Shi-Tomasi corner detector [7], and the Förstner Operator [8]. This paper uses the Harris corner detector to find the RM location that was placed on the store surface, since the criterion to find a corner uses less operations than, for example, the Shi-Tomasi, which has to compute square roots [9].

As described in [9], the simplest possible matching criterion for two image patches is the weighted summed square difference, as described in (1):

$$E_{WSSD}(u) = \sum_{i=1}^{n} w(x_i)[I_1(x_i + u) - I_o(x_i)]^2 \quad (1)$$

Where:

- I_1 and I_0 are two images;
- $u = (u, v)$ is the displacement vector;
- $w(x)$ is a spatially varying weighting function; and
- $\sum_{i=1}^{n}$ is over all pixels in the segment.

Considering small variations in the same image and using a Taylor series expansion of the image function, we have:

$$I_0(x_i + \Delta u) \approx I_0(x_i) + \nabla I_0(x_i).\Delta u \quad (2)$$

Now it is possible to approximate the weighted summed square difference as:

$$E_{AC}(\Delta u) \approx \sum_{i=1}^{n} w(x_i)[I_0(x_i) + \nabla I_0(x_i).\Delta u - I_0(x_i)]^2 \quad (3)$$

$$= \sum_{i=1}^{n} w(x_i)[\nabla I_0(x_i).\Delta u]^2 \quad (4)$$

$$= \Delta_u^T A \Delta_u \quad (5)$$

Where:

- $\nabla I_0(x_i)$ is the image gradient at x_i then:

$$\nabla I_0(x_i) = \left(\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y}\right)(x_i) \quad (6)$$

And the matrix A:

$$A = w \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad (7)$$

Where the weighted summations were replaced by discrete convolutions with the weighting kernel w.

So, the Harris criterion [6] to differentiate interesting key points and localize corners is described in eq (8) with α between [0.04,0.06] and R>>0.

$$R = \det(A) - \propto trace(A)^2 \quad (8)$$

Hardware Architecture

Different kinds of hardware architecture operate in different ways. The work [10] made an overview of the cyber-physical system's architecture concerning machine learning and described the main difference between them.

Up until now, the IPEV has been using the Central Processing Unit (CPU) architecture, as shown in the previous section, which follows the von Neumann or Harvard architecture and uses superscalar pipeline and multi-threading as forms of parallelism [10].

On the other hand, a Field-Programmable gate array (FPGA) consists of an array of programming logic blocks with programming routes between the blocks, which allows a configuration after manufacturing by a designer. The work [10] states that an advantage of an FPGA is the flexibility of refine and replace implementations in the target, and has presented performance over commercial off-the-shelf architectures.

The architectures that mix more than one hardware architecture are known as heterogeneous architecture. An example of this architecture is the System-on-chip that encapsulates a CPU with reconfigurable hardware as an FPGA.

High-Speed Camera

This work uses the high-speed camera EoSens® Creation 2.0CXP2 developed by Mikrotron SVS-Vistek with the capacity of recording 2,247 fps in a 1920 X 1080 pixel resolution (1080p) as a platform of tests. It is a monochrome camera with the CoaXPress high-speed interface which allows real-time sustained transfer rates of up to 12,5 Gb/s.

Such a smart camera provides users access to its internal FPGA (i.e., Kintex UltraScale KU053-2, 2G DDR4).

Besides that, the camera is enclosed in a compact and solid full metal housing robust enough to allow its use in an airborne environment with dimensions of 80 x 80 x 84.6 mm and a C-mount mechanical interface for lens installation.

Reference Marker Tracking Process

The RM tracking (Figure 3) process is composed of 3 steps as follows:

1. Pre-flight operations (i.e., Test setup);
2. Image segmentation to dynamically estimate where each RM is possibly located; and
3. Corner detection.

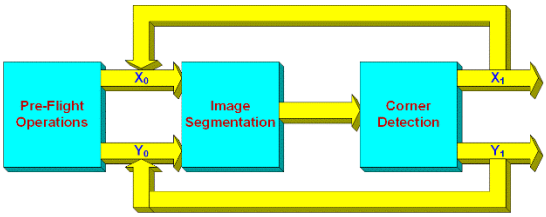

Fig. 03 – 2D Marker Tracking Process

For test setup, it should be executed the following steps:

1. Initially it should be installed several RM's and the ArUco on the store surface and it should be measured their 3D coordinates expressed in the aircraft coordinate reference system with a Total Station;
2. Then it should be computed the camera's intrinsic and extrinsic parameters by using a Stereo Camera Calibration Algorithm [11]. For this step, it should be noticed that the photogrammetric POD translucent window glass also distorts the image. Therefore, to provide better and more accurate results [12], this calibration should be executed with the cameras installed into the POD. Such calibration process:
   a. Computes the camera intrinsics parameters;
   b. Then using as reference, the pinhole camera model, it estimates the optical distortion radial and tangential error minimization compensation parameters; and
   c. It combines the results of all cameras to optimize the solution and to compute the translation and rotation matrices between the master camera and all slaves (i.e., one or more).
3. Finally, it should be determined the translation and rotation matrices between the master camera and the aircraft reference systems.

Once the calibration is performed the position of the RM will be used for tracker initialization, for this reason, it is not possible to move the cameras or the store anymore.

Once the tracking system is initiated the next step is to segment the image framed to define a fixed searching window for corner detection centered in the previously known or initial RM 2D position.

The tracker will search in the subsequent frame for the new position of the RM. Since a high-speed camera is used the new position of the RM will be close to the previous one, which simplifies the tracking process.

The Harris Corner detection algorithm will be used to locate the RM 2D position in the next frame (Figure 4).

The corner detection section initially computes the gradient in x and y of the segmented image and then builds the A matrix as depicted in eq. (7).

To execute as many operations as possible in parallel, the steps in the same stage will be computed in parallel in the FPGA.

This technique can optimize and reduce the execution time in this reconfigurable architecture as compared to a traditional CPU.

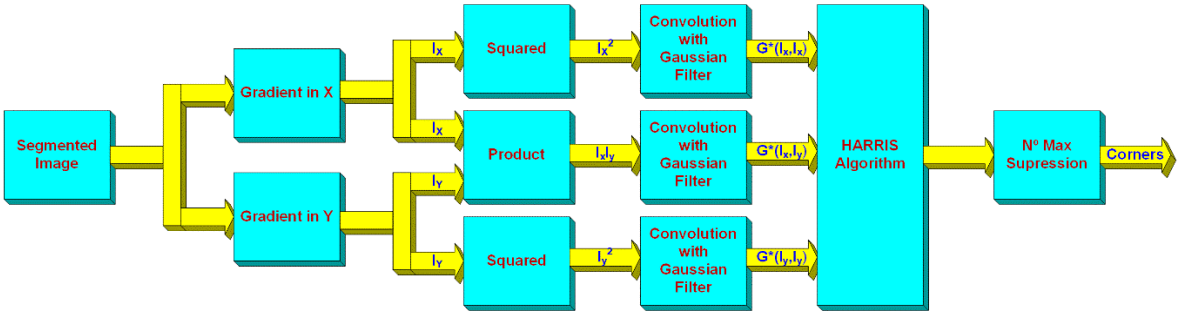

Fig.4. Harris Corner Detection as in [14]

After the computation of the Harris quantity, a non-maximum suppression operation is required to avoid that corner concentration in areas with high contrast.

The new RM position will be used to define the searching window position (i.e., Image segmentation area) updating the tracking.

It is worth mentioning that some works use a prediction step, like a Kalman filter, however, for this application, and since a high-speed camera is used, this step is avoided to reduce the computational load.

Discussion

As mentioned above the two previous flight test campaigns (i.e., SisTrO I and SisTrO II) were satisfactorily executed in conformance with the requirements presented in the Advisory Group for Aerospace Research & Development document AGARD-AG-300-Vol.5 [13].

The errors in the x, y and z axis, measured in the SisTrO I flight test campaign were respectively -0.16 mm ± 0.97 mm@1σ; -0.11 mm ± 1.42 mm@1σ; and -0.26 mm ± 1.21 mm@1σ.

Besides the fact that SISTRO II fulfilled the AGARD-AG-300-Vol.5, requirements it also improves SisTrO I, in some aspects such as faster execution time, and 3D trajectory solution from a single camera for introducing redundancy.

However, a point that has to be observed is that SisTrO I uses an i7 Quad Core @2,4GHz, 8GB DRAM @1600 MHz processor to perform the real-time processing while this present paper designs a solution for an embedded FPGA Kintex UltraScale KU053-2, 2G DDR4 with limited capability as compared to the first one.

Therefore, the CPU architecture used in the SisTrO I achieves better performance than the FPGA architecture.

For this reason, all forms of parallelism and operations optimization should be applied.

Furthermore, the workload to execute the object tracking algorithm and the amount of data that high-speed video produces are the main issues for this work.

Therefore, using FPGA parallelism and Harris corner detector simplifies the tracking processing.

For example, the work in [14] developed a Harris Corner detection algorithm in FPGA aiming for processing high-speed data. The results of the work are presented in Table 1.

Tab. 1: Resource comparison in [14].

Implementation	[14]	[15]	[16]
LUTs	8035	9849	6267
FF	8404	4335	-
BRAM(18K)	17	5	11
BRAM(36K)	0	64	0
DSP	0	0	0
Frame/s	626	144	295

However, the difference between this paper and the referenced one [14] is that in our work the background is considered to be fixed and it was possible to engineer the store with the RM's that simplifies the knowledge of tracking pattern, while theirs tracks natural targets.

In addition, the exact position of the RM's can be accurately measured, and this particular feature speeds the processing algorithm and benefits the real-time tracking performance.

The FPGA Kintex UltraScale KU053-2 contains more hardware resources than was necessary to perform the referenced work [14] (Table 2).

In addition, the camera and its embedded FPGA unit are designed to satisfactory operate in an airborne environment, such as the camera POD.

Tab. 2: Hardware resource of FPGA Kintex UltraScale KU053-2.

Elements	KU053-2	70% availability
LUTs	20328	14229
FF	406256	284379
BRAM(18K)	1080	756
BRAM(36K)	540	378
DSP	1700	1190

Conclusion

This work presented a marker tracking design to execute real-time target tracking in an FPGA inside a small-factor high-speed camera, and it was compared with the two previous flight test campaigns (SisTrO I/SisTrO II) that used a regular embedded computer with a CPU.

The results in the CPU architectures in SisTrO I and SisTrO II showed small errors and the fulfillment of the AGARD-AG-300-Vol.5. Since they have more computational capacity they achieve better performance compared with the reconfigurable architecture that has limited computational capacity and memory.

However, it was seen that an FPGA, with some parallelism and a reduced operations algorithm (e.g. Harris corner detector), can process the object tracking in high-speed video accomplishing the flight test form factor and environmental requirements.

Suggested future works are to execute real flight tests to obtain and compare the results of the RM tracking design, find other algorithms to perform the corner detection or the object tracking, verify the performance of a prediction step like a Kalman filter, use a deep learning algorithm and an FPGA accelerator to perform the inference and perform the 3D trajectory estimation from a single high-speed camera.

Acknowledgment

This work is supported by the funding agency FINEP under the FAEV project (Ref.: 01.22.0545.00).

References

[1] Forsman, Ed & Getson, Scott & Schug, David & Urtz, Greg. (2008). IMPROVED ANALYSIS TECHNIQUES FOR MORE EFFICIENT WEAPON SEPARATION TESTING. DOI:10.13140/RG.2.1.4539.0240

[2] Forni, A. L. C., “Manual de Aerodinâmica, Documento nº 20-R-AH”, Chapter 3, Divisão de Ensaios em Voo, 1995.

[3] JIAA; JIIA CXP-001-2015 CoaXPress Standard, The Standardization Committee CoaXPress Working Group, Japan Industrial Imaging Association (JIIA), 2015, Available at: http://jiia.org/wp-content/themes/jiia/pdf /standard_dl/coaxpress/CXP-001-2015.pdf, accessed on August, 20th, 2018.

[4] Romero-Ramirez, Francisco J.; Muñoz-Salinas, Rafael; Medina-Carnicer, Rafael. "ArUco: a minimal library for Augmented Reality applications based on OpenCV".

[5] Melo, Gabriel Adriano, Marcos Máximo, and Paulo André Castro. "High Speed Marker Tracking for Flight Tests." IEEE Latin America Transactions 20.10 (2022): 2237-2243.DOI: 10.1109/TLA.2022.9885171

[6] Harris, Christopher G. and M. J. Stephens. “A Combined Corner and Edge Detector.” Alvey Vision Conference (1988).DOI:10.5244/C.2.23

[7] Shi, Jianbo. "Good features to track." 1994 Proceedings of IEEE conference on computer vision and pattern recognition. IEEE, 1994.DOI: 10.1109/CVPR.1994.323794

[8] Förstner, Wolfgang, and Eberhard Gülch. "A fast operator for detection and precise location of distinct points, corners and centres of circular features." Proc. ISPRS intercommission conference on fast processing of photogrammetric data. Vol. 6. 1987.

[9] Szeliski, Richard. Computer vision: algorithms and applications. Springer Nature, 2022.DOI:10.1007/978-3-030-34372-9

[10] Loubach, Denis S. "An Overview of Cyber-Physical Systems’ Hardware Architecture Concerning Machine Learning." 2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC). IEEE, 2021.DOI: 10.1109/DASC52595.2021.9594429

[11] ZHANG, Z.; “Flexible Camera Calibration by Viewing a Plane from Unknown Orientation”. In Proceedings of the 7th IEEE Conference of Computer Vision, pp 666-673 vol. 1, DOI: 10.1109/ICCV.1999.791289

[12] de Vasconcelos, Luiz Eduardo Guarino, et al. "Store separation: Photogrammetric solution for the static ejection test." International Journal of Aerospace Engineering 2019 (2019). DOI: 10.1155/2019/6708450

[13] Arnold, R. J., and C. S. Epstein. "Agard flight test techniques series. volume 5. store separation flight testing." Agard Flight Test Techniques (1986).

[14] H. Zhou et al., "A High-Speed Parallel FPGA Implementation of Harris Corner Detection," 2020 IEEE International Conference on Integrated Circuits, Technologies and Applications (ICTA), Nanjing, China, 2020, pp. 71-72, doi: 10.1109/ICTA50426.2020.9332111.

[15] Chao, Tak Lon and Kin Hong Wong. “An efficient FPGA implementation of the Harris corner feature detector.” 2015 14th IAPR International Conference on Machine Vision Applications (MVA) (2015): 89-93.DOI: 10.1109/MVA.2015.7153140

[16] A. Ben Amara, E. Pissaloux, R. Grisel and M. Atri, "Zynq FPGA Based Memory Efficient and Real-Time Harris Corner Detection Algorithm Implementation," 2018 15th International Multi-Conference on Systems, Signals & Devices (SSD), Yasmine Hammamet, Tunisia, 2018, pp. 852-857, doi: 10.1109/SSD.2018.8570491.