ISO/IEC JTC1/SC29/WG1
(ITU-T SG16)

Coding of Still Pictures

JBIG
Joint Bi-level Image Experts Group

JPEG
Joint Photographic Experts Group

TITLE: **Use Cases and Requirements for JPEG XE v2.1**

SOURCE: ISO/IEC SC29 WG1

PROJECT: JPEG XE

STATUS: Final

REQUESTED ACTION: For information

DISTRIBUTION: Public

Contact:
ISO/IEC JTC 1/SC 29/WG 1 Convener – Prof. Touradj Ebrahimi
EPFL/STI/IEL/GR-EB, Station 11, CH-1015 Lausanne, Switzerland
Tel: +41 21 693 2606, Fax: +41 21 693 7600, E-mail: Touradj.Ebrahimi@epfl.ch

Table of contents

1 Introduction 2
2 Scope 3
3 Definitions 3
4 Event-based vision framework 5
4.1 Architecture 5
4.2 Normative components 6
5 Use cases 7
5.1 Embedded edge computing 7
5.2 Mobile devices 8
5.3 Industrial machine vision 9
5.4 Long-term storage 9
5.5 Movie production 10
5.6 Gaming 11
5.7 Scientific and engineering measurement 12
5.8 Multi-sensor TSPI generation 12
6 Requirements 13
6.1 Coding 14
6.2 Systems 14

Use Cases and Requirements for JPEG XE v2.1

1 Introduction

This document describes use cases and their associated requirements for potential standardization of the coding of events in event-based (EB) vision applications, relevant in the field of computer vision and image processing.

Despite the impressive progress during the last decades in the fields of information technology, microelectronics, artificial sensory and information processing, practical systems are still much less effective in dealing with real-world tasks than their biological counterparts. This analysis led to the emergence of the neuromorphic engineering field, and in particular event-based sensing, which aims at building sensing and computing silicon-based devices mimicking how biological systems acquire and process information. Unlike conventional image sensors, EB sensors do not use a common sampling rate – known as the frame rate – for all pixels, but instead each pixel continuously tracks the amount of incident light and asynchronously samples the signal upon change. This highly efficient way of acquiring sparse data, the high temporal resolution, and the robustness to uncontrolled lighting conditions – having high dynamic range – are characteristics of the EB sensing process that make EB imaging attractive for numerous applications, like in industrial automation, process monitoring, surveillance, IoT, AR/VR, automotive and mobile environments.

EB vision technology therefore has an enormous potential, but also brings some challenges regarding its usage and integration. The asynchronous origin of the event data, non-constant output data rates from the sensors, non-standard interfaces, as well as data formats, and in general the way dynamic visual information is encoded inside the event data, pose challenges to the usage and integration of event-based sensor-perception systems in embedded applications.

It is worth observing that major players in the field of EB vision have been mainly focused on defining and developing proprietary interfaces and formats for their products. This severely hinders adoption and interoperability, and might even cause issues regarding security, certification, innovation, etc. In addition, with the advances in semiconductor technology, sensors now typically implement the protocol and (part of the) format encoding in hardware, for efficiency reasons like limiting power usage and latency. This in turn has the drawback that the format cannot be easily changed afterwards, without adding extra complexity again. Moreover, these proprietary solutions are also a potential cause for interoperability issues.

All these difficulties could be significantly improved or solved by the development and adoption of suitable standards that provide a solid framework to work in. Indeed, standardization can deliver a well-defined format to represent visual information captured by EB vision cameras, which will enable interoperability across manufacturers and efficient exploitation by service providers. For these reasons, the JPEG committee thinks that now is the time to pursue an exploration activity to investigate and work towards the standardization of EB formats.

Finally, standardization will allow selecting the best available technology, and avoid re-inventing existing event protocol and format coding technologies. A limited number of relevant standards could most likely cover a wide range of event-based use cases.

An event data format, in addition to providing high compression efficiency, should also offer ultra-low latency and allow for low complexity implementations with low-power consumption. Additionally, with lower priority, the format may also facilitate compressed domain image processing to enhance the quality of the captured visual information, like for example denoising, as well as performing effective computer vision tasks, like object recognition, object tracking, motion segmentation, etc.

This document first provides a scope for standardization, followed by a section that specifies key concepts and definitions in the context of EB vision standardization. Subsequently, a short section gives an overview of a typical EB vision architecture to illustrate and clarify the use cases that follow. Finally, the requirements section and a preliminary timeline follow.

2 Scope

The scope of JPEG XE is the creation and development of a standard to represent *Events* in an efficient way allowing interoperability between sensing, storage, and processing, targeting machine vision and other relevant applications.

3 Definitions

This section defines the key terms and definitions that are used in the context of this document.

Event

An event is the message that signals the result of an observation at a precise point in time. It is typically triggered by a detected change in the physical world. Examples are contrast detection (CD) events from an EB vision sensor or inertial measurement unit (IMU) readings.

Sensor pixel

A sensor pixel, short for “picture element”, is the smallest unit element of an imaging or vision sensor. A sensor pixel of a semiconductor-based image or vision sensor typically consists of a light-sensitive element and other semiconductor elements allowing readout of the light-induced output signal.

Event pixel

An event pixel is a sensor pixel in an event vision sensor that initiates the generation of events in response to a change of light. Like a sensor pixel it consists of a light-sensitive element, with additional analog and digital circuitry for generating the event occurrence signal [3].

Event vision sensor

An event vision sensor is a physical sensor chip that has an array of event pixels, typically organized in a 2-D rectangular grid, like image sensors. An event vision sensor produces a stream of asynchronously generated events. Event vision sensors are also often referred to as “neuromorphic” vision sensors in reference to the fact that they are inspired by the operation of the human retina.

Contrast detection

The act of detecting relative changes in the intensity of incident light (i.e., temporal contrast) exceeding an adjustable threshold by an event pixel.

Contrast detection event pixel

An event pixel that performs contrast detection.

External trigger events

A particular type of event representing a change of external signal and characterized by the ID or address of the external signal, polarity of change of this signal and precise timestamp of this change. Such events are typically useful to timestamp signals from other sensors or hardware using the high-resolution clock of the event vision sensor and enable fine synchronization, for instance in computational imaging applications such as deblurring.

Lossless coding of events

The lossless coding of events refers to the identity equivalence of an event sequence before and after being encoded and decoded, meaning that an input event sequence can be fully and identically reconstructed by the encoding and decoding processes. This identity relationship between two given event sequences is defined by the following two properties:

- individual identity of events: every event from the original sequence has a one-to-one match, with identical type (e.g. contrast detection, trigger) and field values (e.g. timestamp, address, etc), in the decoded sequence and vice versa;
- chronological ordering of events: events in the decoded sequence shall preserve the chronological ordering of the original sequence; however, for multiple events sharing the same timestamp, any spatial ordering is allowed.

As an example, a coding strategy forcing row-major ordering for CD events with the same timestamps, while keeping everything else identical, is a lossless coding strategy independently of the order of events in the original sequence. Alternatively, a coding strategy that alters the chronological order of events between the original and the decoded sequences, while keeping everything else identical, is not considered a lossless coding strategy.

Absolute time

A system of time maintained as traceable to a worldwide reference standard. It includes a definition for time intervals and an epoch with which to reference timestamps. The use of absolute time to mark events enables the correlation of events in time.

Event time

The generation time of an event.

SI Seconds

The International System of Units (SI) base unit for measuring time.

TAI

International Atomic Time is a high-precision continuous scale of time derived from hundreds of precise atomic clocks from around the world and maintained as closely as possible to the International System (SI) second.

TSPI

Time-Space-Position-Information consisting of spatiotemporal data defining location, extent/shape, orientation, and dynamic behavior of one or more objects in 3D space. Includes quality or measurement uncertainty information.

4 Event-based vision framework

4.1 Architecture

Figure 1 shows a reference event-based (EB) vision architecture, illustrating the generation and flow of event data from an array of *event pixels* in an *event vision sensor* to one (of many possible) compute platforms for real-time (or near-real-time) processing in a vision-based application; or to long-term storage for later use of the data, such as data set creation. The *output* from the compute platform contains the application-specific high-level information, extracted from the event data by an algorithm, in a suitable form (like steering angle command, object counting number, optical flow field, and tracking position), and is out of scope for the present standardization effort.

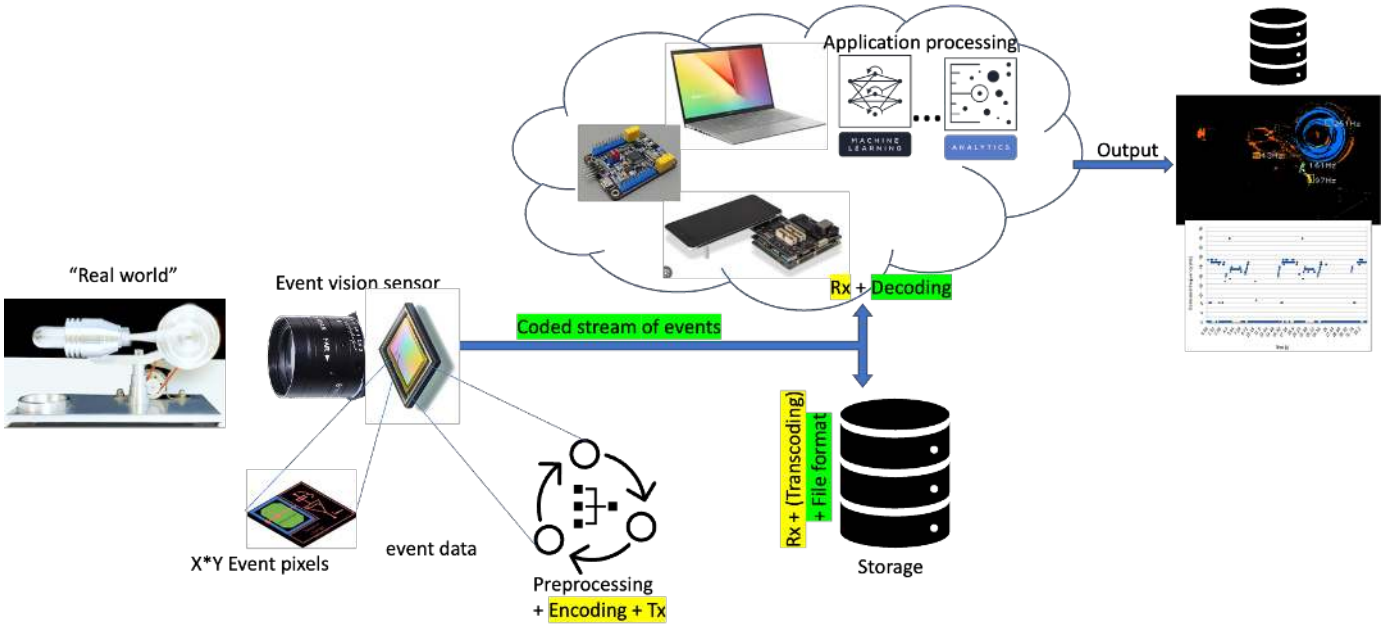

Figure 1 - Illustration of a reference event-based vision architecture. Standardization scope is highlighted in green.

In the following each of the key components of the event-based vision architecture will be explained.

Event vision sensor

In essence, a scene in the *real world* is monitored by an *event vision sensor*. Driven by the dynamics in the scene, each *event pixel* in the *event vision sensor* autonomously and asynchronously generates *events* resulting in a continuous stream of *event data*. Because of the asynchronous nature of the *event* generation, the *event data* stream has a variable instantaneous rate. When there is little variation in the scene, only a small number of events will be generated; when there is strong or fast variation, then substantial amounts of events are generated.

Although in a typical event-based application, the events in the data stream are originating from the array of *event pixels*, an *event vision sensor* can also produce other types of *events* and insert them in the event data stream. More precisely, the *event vision sensor* can monitor and signal other parameters, typically environmental, such as illumination, state of input/output pins, temperature, and inertial measurements, and include these data encoded as *events* (remember *events* are messages, i.e. containers transporting information) into its output event stream.

Event data preprocessing and encoding

The stream of event data generated by the pixel array (and sometimes other functional units, see above) is often manipulated by an on-sensor *Event-based Signal Preprocessing* (ESP) stage, which applies pre-processing steps such as filtering (e.g. noise reduction, edge enhancement, throughput control, etc) and formatting, before the event data are encoded for transmission over the sensor's output data interface. The preprocessing step typically allows making the decision on what event data is relevant and what data can be pruned. Both for lossy and lossless coding, this potential pruning of event data is not considered as loss incurred by the actual coding process. The preprocessing step is not part of the actual coding process. Formats can coarsely be divided into event streaming formats and event aggregation formats (e.g. frames-of-events or 2-D histograms). An *encoding* stage prepares the data for *transmission (Tx)* over a network or transport-link (e.g. MIPI, USB, Ethernet). Typically, *ESP* and *encoding* are integrated on the event vision sensor chip but can also be implemented in off-chip discrete logic.

Transmission (Tx) and reception (Rx)

Transmission and reception are necessary steps to transport the coded stream of events from a source to a destination. Typically transport protocols used for this step are industry standard protocols, like MIPI or DCMI.

Decoding and application processing

At the other end of the transport-link, the *coded stream of events* is then typically *received (Rx)* by a compute platform running the event-based vision applicative software.

The compute platform in this context should be seen in a broad sense and can be one of a server, desktop PC, laptop, embedded microprocessor, GPU, APU, mobile SoC, micro-controller, AI/NN accelerator, SNN processor, etc. The proper platform selection is typically driven by the targeted application and/or market requirements. What is common to any selected platform is that it allows to *receive* and *decode* the coded stream of events for further processing by the application software. Typically, event-based application requirements are low latency, low overall power consumption and low cost to generate the application *output* results.

Transcoding and file format for storage

Instead of immediate application processing, the *coded stream of events* might also be recorded for long term storage and stored into a *file format*. Such stored event-based data can then be replayed later, therefore enabling the exchange of recordings, the creation of massive datasets for AI training, the creation of reference datasets for applicative benchmarks, etc. This could imply transcoding the coded stream of events to emphasize other requirements like stronger compression instead of low latency.

4.2 Normative components

The key normative components for the JPEG XE standardization effort are highlighted in green in figure 1, and are:

- the syntax and semantics of a *coded stream of events* for transport and storage,
- the *file format* used for storage of *coded events*, and
- the *decoding process.*

These will allow for application processing nodes to be developed independently from the *event vision sensor* that generated the *event data*. This in turn helps with cross-vendor interoperability.

Preprocessing and *encoding* operations are outside the scope of the JPEG XE standardization effort because these are not essential for interoperability.

5 Use cases

Many use cases exist in EB vision, characterized by the fact that they typically prefer different compute platforms. Figure 2 illustrates three examples of different compute platforms used in different use cases, and a matching event vision sensor configuration, data format and interface type are used for transmission of the event data to the respective platform. In the following, multiple use cases (or applications), grouped into a number of classes, are presented.

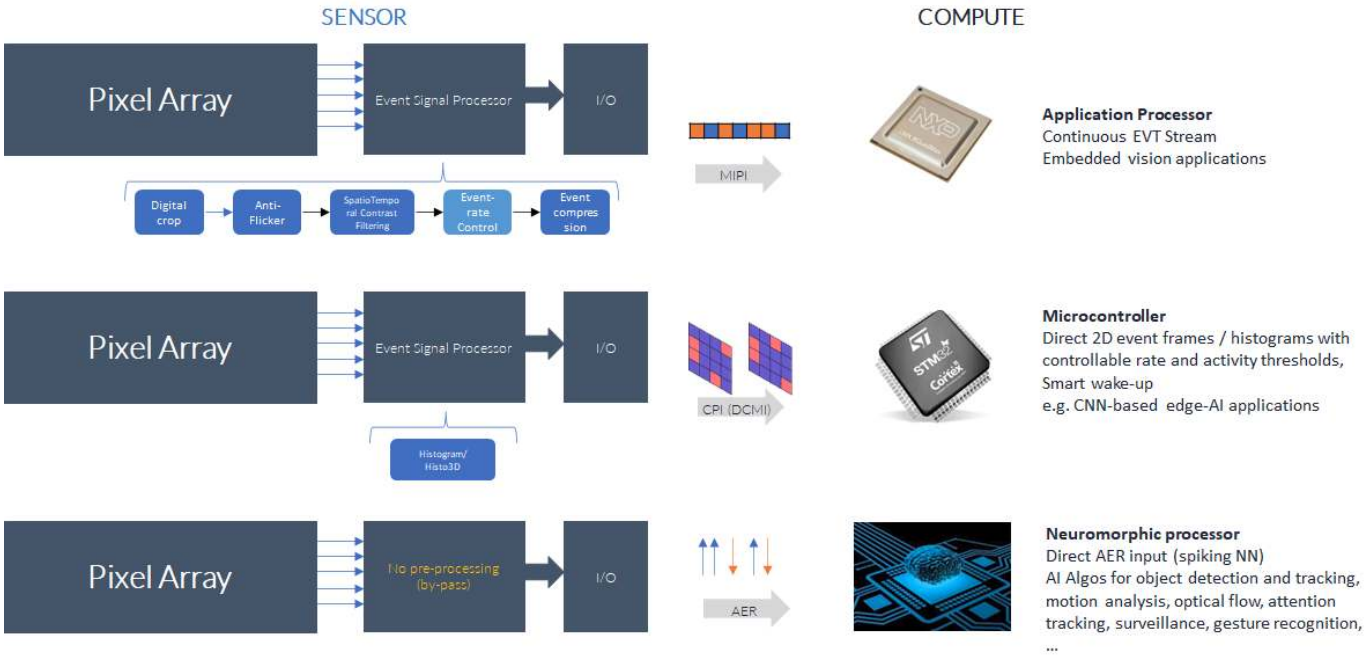

Figure 2 - Illustration of three examples of different compute platforms and their matching sensor configuration, data format and interface type

In the following, a non-exhaustive list of relevant use cases is given, split into seven main categories.

5.1 Embedded edge computing

More and more intelligence is embedded into daily used objects today, as shown by the progress made in IoT with smart appliances, in mini-robotics with drones and vacuum robots, AR/VR headsets, and many more. Event vision sensors allow for efficient and performant visual perception and can therefore be very beneficial in contexts where the perception-processing-action pipeline must run onboard with long autonomy, short reaction time and low overall price.

Typically, the sensor is connected to a microcontroller by means of a Digital Camera Memory Interface (DCMI) compatible low-power low-latency CMOS Parallel Interface (CPI) [1]. Perception resolution can be

small to minimize the necessary processing and secondary information can be used (e.g. external triggers, inertial measurements, global illumination, temperature) to support efficient processing.

Typical compute platforms:

- Microprocessor unit (MPU)
- Micro controller unit (MCU)
- Neuromorphic computing architectures
- AI accelerators

Example applications:

- Eye tracking, localization and mapping & gesture recognition for AR/VR headsets
- Obstacle avoidance & localization and mapping for light-weight mobile robots (e.g. drones, vacuum cleaners)
- Presence detection & action recognition for smart homes & IoT

Relevant features:

- Low power (battery operated)
- Low latency (encoding/decoding supporting real-time processing and closed-loop reactivity)
    - both low latency transmission
    - and low latency processing
- Low (system) cost (because of consumer market)
- Small sensor resolutions
- Insertion of secondary information (external triggers, inertial measurements, global illumination, temperature, etc)

5.2 Mobile devices

Nowadays mobile devices – like mobile phones and tablets that work on SoC processor platforms such as those from QC Snapdragon or MediaTek – are ubiquitous and centralize a huge range of capabilities, used daily by billions of people. As CIS and Time-of-Flight (ToF) sensors in recent years, EB sensors provide a new perception modality which can help robustify, optimize and extend the capabilities of mobile platforms. Data fusion between EB sensors and conventional RGB image sensors can be exploited to significantly enhance the quality of photographs and videos, which is a driving factor in the mobile industry.

The dominant interface to connect sensors with the mobile application processor, is MIPI C-Phy [1]. The application related processing is run on the mobile application processor. Data fusion with conventional RGB image sensors typically requires large sensor resolution (1-2 Mpixels), resulting in high event rates being transmitted to the System-on-Chip (SoC). As such, the event data encoding format needs to enable a high (lossless) compression ratio with efficient decoding, while being compatible with existing MIPI standards [1]. ESP blocks can also be used to aggregate event-based data in preprocessing and prepare it for efficient processing on the application processor. Aggregated event formats can help optimize memory access and minimize the computational effort for the application processor, and hence can take part in reducing the overall system power consumption. Furthermore, secondary information such as external triggers may be needed to effectively synchronize the multiple sensors and perform data fusion. For example, external trigger events can be used in computational imaging applications to precisely signal the exposure start and end timestamps of RGB frames from a conventional sensor in the event vision sensor stream.

Typical compute platforms:

- Mobile processor SoCs (QC Snapdragon, MediaTek)

Example applications:

- Still image deblur, video deblur, super slow-motion for imaging applications
- Augmented reality, gesture & face recognition for sensing applications

Relevant features:

- Support for synchronization with a regular image sensor
- Low power (battery operated)
- Aggregated event data formats
- Insertion of secondary information (external triggers, inertial measurements, global illumination, temperature, etc)
- ISP for image quality metrics

5.3 Industrial machine vision

This category of use cases covers many sensing applications targeting very specific scenarios with the goal of extracting statistics or information to optimize industrial processes. EB sensors provide a new perception modality which can help address such use cases in a faster, cheaper, or more reliable way.

Because of their variety and fragmentation, industrial use cases do not usually share a typical hardware setup; however, some frequent requirements can be identified, even if not always relevant to all use cases.

Typical compute platforms:

- Microcontroller (MCU)
- FPGA
- Generic CPU-based system

Example applications:

- Vibration monitoring
- Object counting, tracking, sorting
- Object scanning (barcodes reading, parcel sizing, …)
- Quality monitoring (welding, surface scratches, parcel damages, …)
- Area monitoring (people / robot trajectories in warehouse, …)

Relevant features:

- Low latency (encoding/decoding supporting real-time processing and closed-loop reactivity)
    - both low latency transmission
    - and low latency processing
- Insertion of secondary information (external triggers, global illumination, temperature, etc)

5.4 Long-term storage

Images and videos are ubiquitous today and standardized file formats largely contributed to the advance of computer vision and artificial intelligence based on RGB images. Standardized file formats for EB data are also needed to further support the advance of computer vision and artificial intelligence based on this new visual modality.

For instance, some applications are inherently non-real time, such as On-Demand Slow Motion where a stream of event data and corresponding RGB video is stored and can later be replayed with any arbitrary and possibly variable slow-motion factor, after the fact. More generally, training artificial intelligence models requires massive datasets, and benchmarking computer vision algorithms requires reference datasets which can be stored for extensive periods of time.

Usage of pre-recorded files is typically most useful on general purpose CPU, by opposition to live streaming and processing which can make sense for any processing platform. To optimize long term storage constraints, data compression is an important factor whose trade-offs will be strongly application dependent and must therefore be configurable. Moreover, metadata related to the acquisition device, settings and conditions are typically useful to integrate into the file for identification after the acquisition. Random access of data stored into the file is typically also useful to enable human inspection of the contents. Efficient decoding can also be important in certain situations, although not always mandatory.

Finally, ESP integrated in the event vision sensor chip may produce aggregated events formats, by opposition to regular event data. The usage of regular event data or aggregated event data will be strongly application dependent, and storing event data directly in aggregated form will save a lot of processing time.

Typical compute platforms:

- Generic CPU-based system

Example applications:

- On-demand Slow Motion
- Artificial Intelligence training datasets
- Computer Vision benchmarking datasets

Relevant features:

- Efficient coding
    - real-time compression of the coded event data for recording of live streams
    - possibility to use lossless coding
    - possibility to use lossy coding with tunable ratio/quality tradeoff
    - possibility for tunable configuration with tradeoff between coding efficiency, resource usage and reconstruction fidelity
- Embedding self-descriptive metadata
- Support for aggregated event data
- Random access to the file contents

5.5 Movie production

EB imaging can be used for high quality content production. In particular, the cameras can be used for motion capture, virtual production, and live capture. The VFX (video effects) and CGI (computer generated imagery) capture process are widely used to capture life-like movement that can be mapped to creatures, mechanical devices or facades giving them anthropomorphic characteristics. Accurate capture of the motion can improve the final rendering of the content. The popularity of virtual sets and virtual productions can also be improved using event-based imaging. The advent of LED walls in virtual production provides a potential mechanism where the movement or lighting changes could trigger localized event capture. These captures can then be used as part of the final compositing. Finally, event-based imaging lends itself to high frame rate capture like that suggested for live sports or live events that are favorable to high frame rate delivery.

Typical application platforms:

- Extremely high-quality camera system
- High Dynamic Range (HDR)capable baseband delivery
- High throughput data paths
- Large storage volumes

Example applications:

- Motion capture
- Virtual production
- Live capture

Relevant features:

- Multi-camera intelligent capture
- High sensor resolutions (4k - 8k)
- HDR and Wide Color Gamut (WCG)
- High data storage
- Post-production workflows

5.6 Gaming

Computer vision and special effects based on video analytics tools are widely used in gaming applications. Event based cameras can become efficient alternatives for consumer and professional gaming applications. Event-based cameras tend to provide gesture recognition without suffering from limitations such as motion blur and latency, especially in situations where the subject must perform abrupt and quick movements which is typical in many gaming scenarios. The low power consumption of event-based cameras can become assets for tools that need to be autonomous while offering a compact form factor.

Typical application platforms:

- PC and desktops
- Servers on the cloud
- Smartphones and tablets
- Gaming consoles

Example applications:

- Motion capture
- Gesture recognition
- Game controllers
- Visual input interfaces

Relevant features:

- Low power consumption
- Low complexity
- Ultra-low latency

5.7 Scientific and engineering measurement

This category of use cases covers the use of event sensing applications to perform measurements for scientific or engineering purposes. EB sensors provide extreme low latency and can detect changes in extreme lighting conditions that are not feasible using common vision sensors. Moreover, the variable event rate at the output, driven by actual changes in the observed scene, allow making long recordings without the need to store or process non-relevant information.

Typical compute platforms:

- Microcontroller (MCU)
- FPGA
- Generic CPU-based system
- GPU platform

Example applications:

- Observation of lightning strike propagation (from earth and from space)
- Observation of lunar eclipse for scientific study
- Monitoring of space rocket launches
- Monitoring of “falling stars” and debris entering the atmosphere
- Space Situational Awareness (tracking satellites and space debris)

Relevant features:

- Absolute time base (to support multiple synchronized cameras)
- Ultra-low latency
- Variable event rates
- HDR and WCG
- High sensor resolutions
- Support for metadata

5.8 Multi-sensor TSPI generation

Use cases requiring the correlation of different events from different sensors require an ability to label the events with a common source of time. While a common, local source of time can be used to correlate events from sensors located close to each other, the use of a globally available, absolute time scale enables the correlation of events from distant locations. A common use case involves generating time-space-position-information (TSPI) for a scene object by making measurements from multiple independent sensors at different locations. Triangulation calculations involving event data from multiple sensors determine the precise location and orientation of scene objects of interest. This process generally involves triangulation where the line of sight to a scene object in the real world is determined using event data, as well as platform gimbal angles, the sensor location, and other relevant metadata. Figure 3 depicts a use case scenario where absolute timestamps are required to facilitate calculations.

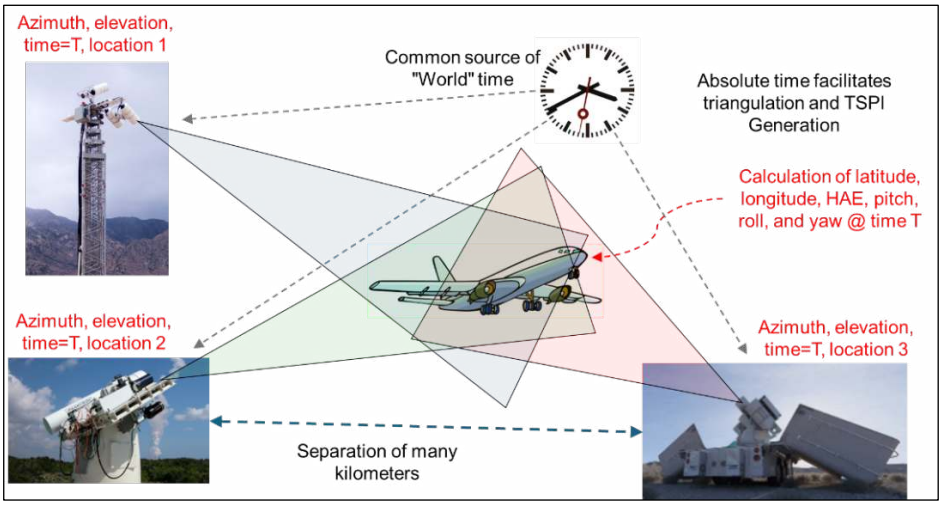

Figure 3: Triangulation and TSPI generation use case scenario.

Typical compute platforms:

- Separately located, networked, real-time processing-based systems
- Post capture data processing/analysis on general CPU systems

Example applications:

- 3D object tracking, especially from large distances with sensors at multiple locations
- Scene object location, shape, size, and orientation determination
- Scene object dynamics in 3D – velocity, acceleration, jerk, etc.
- Overlaps heavily with scientific and engineering measurement use cases

Relevant features:

- Absolute time base (to support multiple synchronized cameras)
- Capture of absolute timestamps at the sensor (to support high accuracy)
- Labeling of events in a manner allowing for the traceability to absolute time standard.
- Support for absolute timing metadata

6 Requirements

Based on the use cases, two categories of requirements have been identified:

- Coding
- Systems

The following sections define in detail the requirements based on the needed functionality for the use cases presented above.

While the ‘shall’ requirements refer to mandatory/essential requirements for the standard, i.e. the standard must include tools to fulfill those requirements, the ‘should’ requirements refer to desirable/complementary requirements. Naturally, at running time, the users will just use the tools that they find needed.

6.1 Coding

- R1.1 The standard shall support the representation of contrast detection **event elements** including, but not limited to:
    - x- and y-coordinates of the event pixel (sensor array address, supporting different sensor array sizes).
    - event polarity (light-to-dark, or dark-to-light).
    - event timestamps with direct relation to absolute time base (with a time resolution matching or better than the temporal resolution of the event generator, e.g. the event pixel).
    - optional absolute timestamps provided at a selectable data packet interval, and linked to the event timestamps, to ensure event timing is traceable to a common absolute time reference with very high levels of accuracy.
- R1.2 The standard shall allow for very **low algorithmic end-to-end coding latency** (for individual events) for live event streams.
- R1.3 The standard shall support **lossless coding** with a high compression ratio.
- R1.4 The standard shall support **different spatial and temporal resolutions and granularities.**
- R1.5 The standard shall support **external event types** that are not originating from the EB pixels.
    - shall support external trigger events.
    - should support other types of user-specific/custom events.
- R1.6 The standard shall support **low power** and **low complexity** tools for live event streams.
- R1.7 The standard shall support **random access** for navigation (decoding at arbitrary points in an event stream) and recovery from data loss in live streams.
- R1.8 The standard shall allow for **efficient coding and memory organization of event data**.
- R1.9 The standard shall support coding of common **aggregated event formats** like 2-D histograms.
- R1.10 The standard shall support **lossy coding** with a tunable compression ratio to further reduce the required data rate in comparison to lossless coding.
- R1.11 The standard should support **scalable representation** in the resolution and time dimensions.
- R1.12 The standard should support **region of interest** functionality.
- R1.13 The standard should be **resilient to data loss**.
- R1.14 The standard should support lossless coding with a **tunable tradeoff** between compression ratio and complexity/latency.

6.2 Systems

- R2.1 The standard shall define a **file format** with the following characteristics:
    - self-contained file (self-descriptive metadata).
    - support for storage of the metadata.
    - support for random access (enable easy navigation through time of the data).

  - support for the inclusion of static absolute time source (clock) metadata and high resolution, absolute timestamps and timestamp quality metadata for each timed sample (which is a defined block of timed event data).

- R2.2 The standard shall support **direct storage of a live event stream** into the file format.
- R2.3 The standard shall support the **representation of metadata** such as manufacturer, sensor model, sensor resolution, description, acquisition date & time, and acquisition parameters.
- R2.4 The standard shall support the **representation of periodic event-related metadata** like frame counters and event counters.
- R2.5 The standard shall support storage of common **aggregated event formats** like 2-D histograms.
- R2.6 The standard shall support **storage of synchronized multi-modal data** such as image frames along with events.
- R2.7 The standard should **allow for easy adaptation to industry standard protocols**.
  - facilitate packing in protocols like DCMI [2] or MIPI [1].
  - support frame padding (to meet packet size constraints).
  - support frame-size matching.
- R2.8 The standard shall support:
  - event data labeling with absolute timestamps in a universally accepted timeframe (TAI, etc.) with a universally traceable epoch (e.g., not based on a relative time, such as camera power-on, etc.).
  - the carriage of absolute time in a system using SI seconds as the base time unit.
  - the format of the timestamp shall allow to compute a difference in time between any two measurements or events (i.e. robust against roll-over).
  - support for nanosecond timestamp resolution, as well as more coarse resolutions (microseconds, milliseconds, etc.) depending on application needs.
  - support for clock source and timestamp metadata indicating source, status, and quality of timestamps.
  - interoperability with common media file formats.

References

[1] MIPI Alliance, https://www.mipi.org/.

[2] Digital Camera Memory Interface (DCMI), AN5020: “Introduction to digital camera interface (DCMI) for STM32 MCUs”, ST Electronics, https://www.st.com/resource/en/application_note/an5020-digital-camera-interface-dcmi-on-stm32-mcus-stmicroelectronics.pdf, Aug 2017.

[3] Lichtsteiner, C. Posch and T. Delbruck, "A 128× 128 120 dB 15 µs Latency Asynchronous Temporal Contrast Vision Sensor," in IEEE Journal of Solid-State Circuits, vol. 43, no. 2, pp. 566-576, Feb. 2008.