

INTEGRATION OF CHRONOS HIGH-SPEED CAMERAS INTO THE MERCURYRT DIGITAL IMAGE CORRELATION SYSTEM

Martin Kmenta¹

¹Department of Economics, Faculty of Business and Economics, Mendel University in Brno, Zemědělská 1, 613 00 Brno, Czech Republic

DOI: https://doi.org/10.11118/10.11118/ecos2026_013

ABSTRACT

This article deals with the design and implementation of an integration module that connects Chronos 1.4 and 2.1-HD high-speed cameras with the MercuryRT digital image correlation (DIC) system. The work responds to the previously missing integration that required manual transfer of recordings and separate setup of both systems. The proposed solution uses the camera REST API for control and an RTSP stream decoded by an external FFmpeg process for live preview. A custom integer-based YUV420p-to-RGB conversion algorithm in C# was implemented; it handles a 10 FPS preview in real time with typical latency around 300 ms. Functionality and stability were verified by a four-hour stress test and trigger measurements using an LED flash; the recording completed without frame loss.

Keywords: high-speed imaging, digital image correlation, REST API, RTSP, FFmpeg

JEL Code: C88

1 INTRODUCTION

Digital image correlation (DIC) is a contactless method for surface deformation measurement that obtains full-field deformation maps from a pair of images captured before and after loading, replacing point-based sensors. Sutton *et al.* (2009) summarized the mathematical foundations of the method; later studies applied it to crack monitoring in wood (Tippner *et al.* 2019), in sheet metal materials (Kořeček *et al.* 2022) and to volume change measurement using 2D DIC (Kotlánová *et al.* 2023). Combining DIC with a high frame rate enables analysis of fast events — from component vibration (Beborniss and Ehrhardt 2016) to dynamic failure. The cost barrier of high-speed imaging was lowered by the low-cost CMOS camera design of Duke *et al.* (2018, 2019), which forms the basis of the Chronos series from the Canadian company Kron Technologies.



This article is licensed under CC BY-SA 4.0 Attribution-ShareAlike 4.0 International license.
<https://creativecommons.org/licenses/by-sa/4.0/deed.en>

The DIC laboratory at the Department of Informatics, Mendel University, uses the MercuryRT software by Mercury MS s.r.o. Version 3.1.7 supports a range of industrial cameras (Basler, FLIR, Photon), but no integration module existed for Chronos. Without it, the user had to switch between the camera interface and the application and transfer frames manually via a network share. The aim of this work was to remove this gap and enable camera control directly from the environment where the results are evaluated.

1.1 Chronos Communication Interface

The Chronos camera contains an embedded Linux computer and exposes a REST API version 0.7.0 via the /control path. Fielding (2000) described the architecture of HTTP-based web services in his dissertation; this is the conceptual basis of REST used by the camera API. For preview the camera provides an RTSP stream defined in RFC 2326 (Schulzrinne *et al.* 1998) in H.264 format (Richardson 2011). The stream is encoded in the YUV420p pixel format, which separates the luminance component Y from the chrominance components U, V and stores colour at half resolution; this saves bandwidth but requires conversion to RGB for most computer vision tasks.

1.2 MercuryRT Modularity

MercuryRT is built around the abstract CameraBase class, which defines the minimum interface for any connected camera (IsConnected, OpenCamera(), CloseCamera(), NewFrame). High-speed devices add the IHighSpeedCamera interface with methods for sequential recording

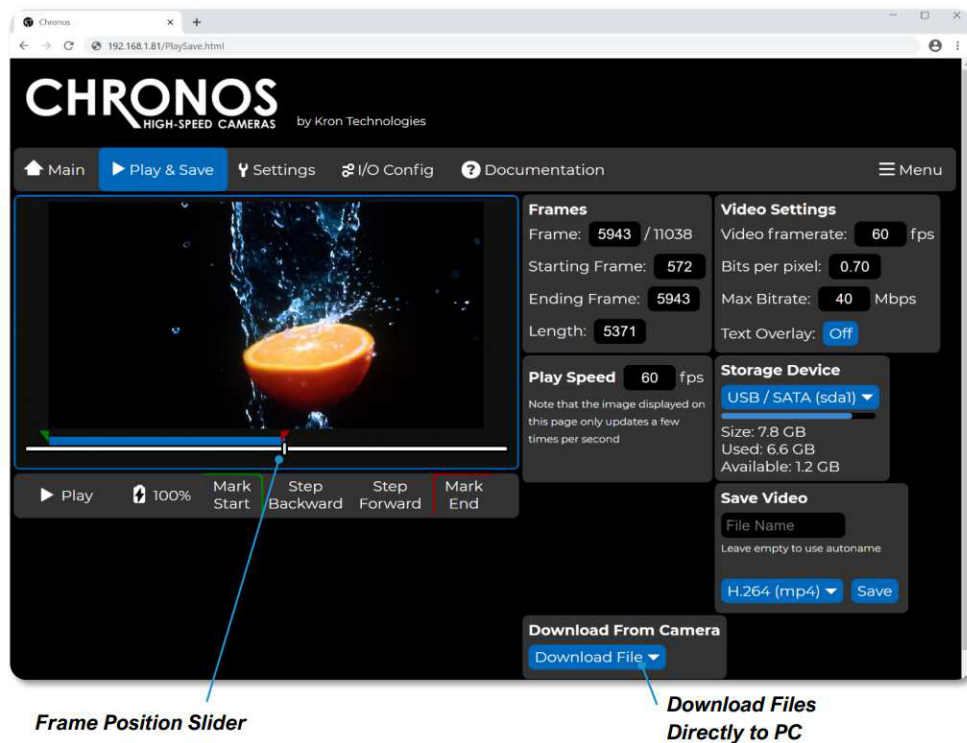


Fig. 1: Chronos web user interface — the Play & Save tab for selecting the frame range, output format and recording download; the left panel shows the live preview of a high-speed scene

(ReadyCameraRecording(), TriggerCameraRecording(), RetrieveHsFrame()). A class for a new camera must implement both interfaces — this automatically integrates the new type into the existing pipeline (probes, storage, correlation analysis) without touching the application core.

1.3 Integration Module Architecture

Three cooperating classes were designed for Chronos. ChronosCamera inherits from CameraBase and implements IHighSpeedCamera; its constructor accepts the IP address and verifies camera availability by querying the model and serial number. ChronosCameraAPI wraps all REST calls (getRequest(), postRequest()) and a Server-Sent Events (SSE) listener for asynchronous notifications about recording completion. ChronosCameraGrabber runs in its own thread, launches an external FFmpeg process, reads raw YUV420p data from its stdout and converts them to RGB frames. The control channel (REST) is thus separated from the data channel (RTSP) — each runs independently and a failure in one does not block the other.

The recording sequence differs from cameras that stream every frame in real time. Chronos records into internal RAM and only allows download after the sequence ends: (i) the user sets parameters, (ii) the startRecording call starts a circular buffer, (iii) after the trigger the camera completes the recording once post-trigger frames are filled, (iv) startFilesave converts data from RAM to a file (MP4/H.264, DNG, TIFF) and (v) the MercuryRT Capture Tool offers HTTP download and immediate import into the project.

2 REAL-TIME YUV420P TO RGB CONVERSION

The YUV420p pixel format stores one byte of luminance Y per pixel while the U and V chrominance components are shared by four neighbouring pixels (4:2:0 subsampling); the total data volume is 1.5 bytes per pixel. For a resolution of 1280 × 1024 at 60 FPS this means about 120 MB/s — naive iteration with floating-point arithmetic proved insufficient. The implemented algorithm went through three optimization steps: (i) conversion coefficients (1.164, 1.596, 2.017) were stored as constants, (ii) recomputed to integers by scaling with a factor of 2⁸=256 (multiplication by 1.164 was replaced by multiplication by 298 with a subsequent right shift by 8 bits), (iii) the loop was placed in an unsafe block with pointer arithmetic. The resulting conversion is:

$$\begin{aligned} Y' &= 298 \cdot (Y - 16) + 128 \\ R &= (Y' + 409 \cdot (V - 128)) \gg 8 \\ G &= (Y' - 100 \cdot (U - 128) - 208 \cdot (V - 128)) \gg 8 \\ B &= (Y' + 516 \cdot (U - 128)) \gg 8 \end{aligned} \tag{1}$$

Results are clamped to the 0-255 range. The grayscale version for DIC modules is computed as a weighted sum $0.299R + 0.587G + 0.114B$ for compatibility with the other cameras supported by MercuryRT.

External RTSP decoding is handled by FFmpeg launched with parameters `-r 10 -f image2pipe -pix_fmt yuv420p -vcodec rawvideo -`. Limiting the rate to 10 FPS prevents overloading the decoder; higher frame rates produced increased latency and artifacts. The first SSE reading variant using a long-running `HttpRequest` proved unstable — the robust implementation waits in a loop for event: and data: lines and re-establishes the connection if interrupted.

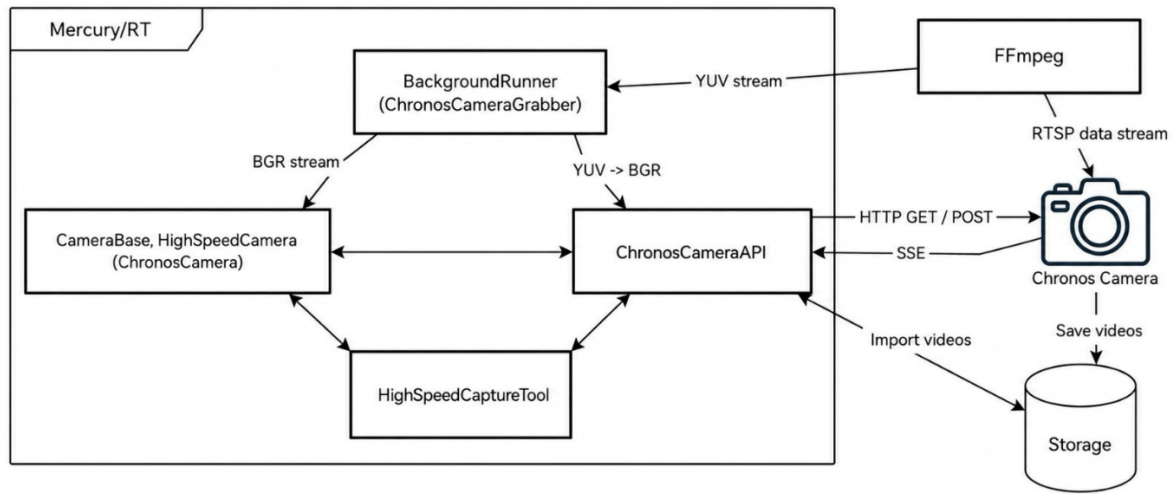


Fig. 2: Data flow between the Chronos camera, the FFmpeg decoder and the MercuryRT modules — RTSP stream for YUV frames and HTTP/SSE for REST API control and video download

3 RESULTS

Live preview latency was measured by two methods: a digital stopwatch displayed on a monitor that the camera filmed, with parallel capture by a reference camera. The time difference was approximately 300 ms at a 10 FPS preview. Attempting to raise the rate to 30 FPS led to latency on the order of seconds and frame skipping caused by FFmpeg decoder saturation — the CPU could not keep up with 30 FPS conversion, as a counter test showed: at 10 FPS the system processed 595 frames out of an expected 600 in 60 s, while at 30 FPS the losses reached tens of percent. The frame rate was therefore fixed at 10 FPS for stability (Kmenta, 2026).

Recording reliability was verified by a trigger → save → import test. The camera was set to 1000 FPS at 640 × 480 resolution with a 5-second pre-trigger; a brief LED flash was generated in the field of view. The recording captured the flash exactly in the expected frame, confirming both the pre-trigger buffer function and correct synchronization. To verify memory stability the preview ran for 4 hours with short recordings every 30 minutes. Application memory usage remained stable with a minimal negligible increase matching a comparative run with a Basler camera — the small increases stem from the MercuryRT core, not from the Chronos module. Tab. 1 summarizes the results.

Tab. 1 Summary of integration module testing results

Metric	Value	Note
Preview latency (10 FPS)	~300 ms	Including decoding and YUV-to-RGB
Throughput (10 FPS)	595/600 frames in 60 s	0.8% loss
Latency at 30 FPS	seconds	Not recommended
4-hour stability	no memory growth	Matches Basler camera
Trigger detection	no frame loss	LED flash test

4 DISCUSSION

Separating REST and RTSP into distinct classes proved valuable in practice — control communication remained stable even during debugging of the data channel and vice versa. This approach is suitable for similar integrations of high-speed cameras that expose both interfaces simultaneously but lack an official C# SDK. The main limitation is decoding performance: even after three rounds of optimization the C# code reliably handles YUV-to-RGB conversion only up to 10 FPS on a typical development machine. For 30 FPS it would be necessary to parallelize the loop across multiple threads, use SIMD vector instructions, or rely on the specialized libyuv library at the cost of a native DLL dependency. For the purposes of a live preview — which serves only for setting up the experiment, not for the recording itself — current performance is sufficient.

5 CONCLUSIONS

The work demonstrated that commercially available Chronos high-speed cameras can be fully integrated into the existing MercuryRT DIC system using standard protocols (REST + RTSP) and an external FFmpeg decoder, without waiting for an official SDK from the manufacturer. The key was an architecture separating the control channel from the data channel and a custom integer-based YUV-to-RGB conversion in C# with an unsafe block. Three results confirmed usability: a stable preview with 300 ms latency, lossless pre-trigger recording verified by the LED test, and memory stability over a four-hour run. Future expansions may include GPU acceleration of the YUV-to-RGB conversion (CUDA, OpenCL) to raise the preview rate towards the native 60 FPS of the RTSP stream, and structured logging of REST calls for post-mortem analysis.

Acknowledgements

This paper was supported by the project CZ.02.1.01/0.0/0.0/16_017/0002334 Research Infrastructure for Young Scientists, which is co-financed from the Operational Programme Research, Development and Education.

REFERENCES

- BEBERNISS, T. J., EHRHARDT, D. A. 2016. High-speed 3D digital image correlation vibration measurement: Recent advancements and noted limitations. *Mechanical Systems and Signal Processing*. 86, 35-48. ISSN 0888-3270. <https://doi.org/10.1016/j.ymssp.2016.04.014>
- DUKE, D. J., KNAST, T., EDGINGTON-MITCHELL, D. 2018. Low Cost High Speed Digital Cameras for Experimental Fluid Mechanics. In: *Proceedings of the 21st Australasian Fluid Mechanics Conference*. Adelaide, Australia.
- DUKE, D. J., KNAST, T., THETHY, B., GISLER, L., EDGINGTON-MITCHELL, D. 2019. A low-cost high-speed CMOS camera for scientific imaging. *Measurement Science and Technology*. 30(7), 075403. ISSN 1361-6501. <https://doi.org/10.1088/1361-6501/ab1832>.
- FIELDING, R. T. 2000. *Architectural Styles and the Design of Network-based Software Architectures*. PhD Dissertation. University of California, Irvine. <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

- KMENTA, M. 2026. *Integration of Chronos high-speed cameras into the MercuryRT system* [in Czech]. Master's thesis, head: Farana, R. Brno: Mendel University in Brno.
- KOŘEČEK, D., SOLFRONK, P., SOBOTKA, J. 2022. Analysis of the Dual-phase Steel DP500 Stress-strain Characteristics During the Plane Shear Test. *Manufacturing Technology*. 22(1), 34-38. ISSN 1213-2489. <https://doi.org/10.21062/mft.2022.015>.
- KOTLÁNOVÁ, B., JAVOŘÍK, J., ŽALUDEK, M., OGUNLEYE, R. O. 2023. Volume Change Measurements Using 2D DIC System. *Manufacturing Technology*. 23(3), 307-312. ISSN 1213-2489. <https://doi.org/10.21062/mft.2023.044>.
- RICHARDSON, I. E. G. 2011. *The H.264 Advanced Video Compression Standard*. 2nd ed. Chichester: John Wiley & Sons. ISBN 978-0-470-51692-8.
- SCHULZRINNE, H., RAO, A., LANPHIER, R. 1998. *Real Time Streaming Protocol (RTSP)*. RFC 2326. Internet Engineering Task Force. <https://datatracker.ietf.org/doc/html/rfc2326>
- SUTTON, M. A., ORTEU, J.-J., SCHREIER, H. W. 2009. *Image Correlation for Shape, Motion and Deformation Measurements: Basic Concepts, Theory and Applications*. New York: Springer. ISBN 978-0-387-78746-6.
- TIPPNER, J., PRAUS, L., BRABEC, M., SEBERA, V., VOJÁČKOVÁ, B., MILCH, J. 2019. Using 3D digital image correlation in identification of defects of trees subjected to bending. *Urban Forestry & Urban Greening*. 46, 126513. ISSN 1618-8667. <https://doi.org/10.1016/j.ufug.2019.126513>.

Contact information

Martin Kmenta: e-mail: martin-kmenta@seznam.cz