WHITE PAPER

Glass-to-Glass Latency: How to Measure and Reduce Video Delay in Medical and Industrial Systems

Where video delay comes from, how to measure it, and how to design it out.

Enciris Technologies

Version 1.0, November 2026

https://www.enciris.com/news/glass-to-glass-latency-white-paper/

Glass-to-glass latency is the time between light entering a camera lens and the same image leaving the glass of a display. In a surgical tower, an inspection cell or a remote-controlled vehicle, it decides whether the person or the algorithm watching the screen can act on what is happening now, or on what happened a moment ago. This white paper explains where the delay comes from, stage by stage, how to measure it reliably, and which design choices remove the most of it. It ends with a latency budget calculator you can use on your own system.

> **Key takeaways**
>
> - Glass-to-glass latency is the sum of every stage between the lens and the display. At 60 fps, every frame buffered anywhere in the chain costs 16.7 ms.
> - Line-based capture and overlay add fractions of a millisecond; frame-based stages add at least one frame each.
> - Encoder defaults can add more delay than everything else combined: a 40-frame lookahead is 667 ms at 60 fps [10].
> - Telesurgery studies rate 200 ms or less as ideal [3] and 100 ms or less as acceptable [4]; local surgical video should be well below.
> - Measure end to end with an optical method and report the median, 95th percentile and maximum.

What is glass-to-glass latency?

Researchers at the Technical University of Munich define glass-to-glass (G2G) delay as the time from the moment the photons of a visible event pass through the lens of a camera until the corresponding photons leave the display [1]. It is an end-to-end figure: it includes everything in between, from the image sensor to the last pixel of the monitor.

Two related figures are useful to keep apart. **Glass-to-algorithm (G2A)** latency stops when the image is available to a processing algorithm, for example an AI model detecting an instrument or a defect [2]. **Motion-to-photon** latency, used in virtual reality, starts with a movement instead of a camera image. In a machine-vision loop G2A is what limits how fast a robot can react; in a system watched by people, G2G is what they feel.

Latency is not the same as frame rate. A 60 frames per second system can still show every frame 200 ms late. Frame rate decides how smooth motion looks; latency decides how far behind reality the picture is. Both matter, but they are engineered differently.

Because almost every stage of a video pipeline works in frames or in lines, it helps to think of latency in those units first, then convert to milliseconds:

Frame rate	One frame	One line at 1080p	One line at 2160p (4K)
25 fps	40.0 ms	37.0 µs	18.5 µs
30 fps	33.3 ms	30.9 µs	15.4 µs
50 fps	20.0 ms	18.5 µs	9.3 µs
60 fps	16.7 ms	15.4 µs	7.7 µs
120 fps	8.3 ms	7.7 µs	3.9 µs

Duration of one frame and one line at common frame rates (active lines; blanking makes a real line period slightly shorter).

The table shows why the difference between a frame-based and a line-based design is so large: at 60 fps one buffered frame costs 16.7 ms, while a buffer of 20 lines of a 4K picture costs about 0.15 ms, more than a hundred times less.

Why video latency matters: surgery, industry and remote operation

Surgery and interventional imaging

In endoscopy, laparoscopy, microscopy and robotic surgery the surgeon's hands move while their eyes are on a screen. Every millisecond of delay separates the movement from its visible result. The most detailed data comes from telesurgery research. In a study on the dV-Trainer robotic simulator, sixteen subjects performed dissection and needle-driving tasks with delays from 0 to 1,000 ms: task time, instrument motion and errors grew exponentially with latency. The authors concluded that latencies of 200 ms or less are ideal, 300 ms is also suitable, 400–500 ms may be acceptable but already tiring, and 600–700 ms is only acceptable for low-risk, simple procedures [3].

A 2022 study with a surgical assist robot and 34 participants set a stricter limit: communication delays may be acceptable if they are 100 ms or less [4]. It also found that surgeons experienced in robotic surgery, working with a delay of up to 100 ms, performed as well as or better than less experienced participants working without any delay: latency is one factor among many, not the whole story.

These thresholds were measured for remote surgery, where the network is the dominant delay. In a local operating room the expectation is tighter, because surgeons compare the screen with a direct view and with the feel of their instruments. The practical goal for local surgical video is to keep the whole chain within a few frames, and the parts you control within a frame or less.

Industrial inspection, robotics and machine vision

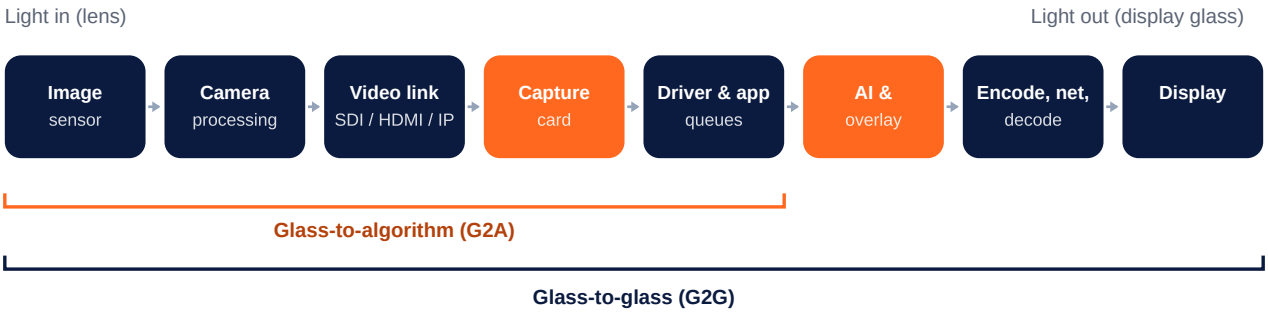
On a production line the question is not comfort but throughput and accuracy. If a reject gate must fire 40 ms after a part passes the camera, the glass-to-algorithm latency plus the decision time must fit in those 40 ms, every time. In closed-loop robotics, delay in the vision path acts like delay in any control loop: it reduces the gain you can use before the system oscillates. Bachhuber et al. showed that a carefully designed pipeline built from off-the-shelf hardware can reach average G2G and G2A delays of 21.2 ms and 11.5 ms for VGA video [2], far below what default settings give.

Remote operation and defense

Drivers of remotely operated vehicles, drone pilots and operators of remote weapon stations or cranes all close the loop through video. Here latency combines with the network: a link that adds 50 ms each way doubles the delay the operator perceives when a command is sent and its effect is observed. The same engineering rules apply: remove buffering, keep compression low-delay, and measure the end-to-end figure, not only the parts.

Where the delay comes from: the latency chain

A video system is a chain of stages, and the end-to-end latency is the sum of the delay added by each one. The diagram below shows a typical chain. Not every system has every stage: a direct-view surgical display may stop after the capture card, while a streaming system adds encoding, a network and decoding.



The latency chain. Orange stages can be line-based in hardware (capture, overlay); every other stage can add a frame or more if it buffers. Free to reuse with a link to this page.

The rest of this section goes through the stages in order. For each one, the useful question is the same: does it work on lines (sub-frame), or does it wait for a whole frame before passing it on?

1. Image sensor: exposure and readout

A camera cannot output a line before it has been exposed and read out. With a rolling-shutter CMOS sensor, lines are read one after another over most of the frame period, so the last line of a frame leaves the sensor almost one frame after the first. Long exposure times add to this. Higher frame rates shorten both, which is one of the simplest ways to reduce latency when the light allows it.

2. Camera image processing

Demosaicing, noise reduction, colour correction and scaling can be done on a few lines at a time, or on whole frames. Temporal noise reduction and some stabilisation algorithms need previous frames and usually hold at least one. Camera datasheets rarely give this figure, so it is worth measuring the camera alone, from lens to output connector.

3. Video link: SDI, HDMI or IP

Uncompressed baseband links such as SDI and HDMI add almost no delay by themselves: pixels travel as they are produced. SDI covers the formats used in professional and medical video, from 270 Mbit/s

SD-SDI (SMPTE ST 259) to 1.485 Gbit/s HD-SDI (ST 292), 2.97 Gbit/s 3G-SDI (ST 424) and 11.88 Gbit/s 12G-SDI (ST 2082), which carries 4K at 60 fps on a single coaxial cable [6]. HDMI 2.1 Ultra High Speed cables support systems up to 48 Gbit/s, and the HDMI 2.2 specification raises this to 96 Gbit/s [7]. Delay appears when something in the path re-times, converts or scales the signal: a converter that buffers a frame adds a frame.

4. Capture card or frame grabber

This is where designs differ most. A frame-based capture card stores a complete frame in its memory, then transfers it to the host: at least one frame of delay before the software sees anything. A line-based design writes the picture to host memory as the lines arrive, so the first lines are available while the rest of the frame is still coming in. Hardware functions such as loop-through outputs and overlays can also be line-based; Enciris specifies a combined loop-through and overlay delay of less than 20 lines of video for its 4K products [13].

5. Driver and application buffering

Operating-system video APIs work with queues of buffers: the driver fills buffers in order and the application takes them in order [12]. A queue protects against dropped frames when the software is briefly late, but if it fills up, every waiting frame adds one frame time. Applications that always take the newest frame, drop late ones and keep queues short avoid hidden delay; applications that process every frame in order can drift several frames behind without anyone noticing.

6. Processing, AI inference and overlay

Inference time is easy to measure and usually well known: a detection model might take a few milliseconds on a GPU or an edge module. The hidden cost is around it: copying frames between memories, converting colour formats, waiting for a batch to fill, and rendering the result. Drawing the overlay in the video path itself, in hardware, avoids sending the picture through a graphics stack and back.

7. Encoding, network and decoding

Compression is the stage most likely to add large, avoidable delay. Encoders reorder frames when they use B-frames, and many look ahead at future frames to decide how to spend bits. With its defaults, the widely used x264 encoder uses 3 B-frames and a 40-frame rate-control lookahead; its `zerolatency` tune sets both to zero and switches to sliced threads [10]. At 60 fps, a 40-frame lookahead alone is 667 ms. Hardware encoders offer the same trade-off: FFmpeg's NVENC encoders, for example, have a `zerolatency` option that requests operation with no reordering delay [11].

Where bandwidth allows, uncompressed or lightly compressed transport avoids the problem: SMPTE ST 2110 carries uncompressed video over managed IP networks [8], and JPEG XS (ISO/IEC 21122) was designed for an end-to-end latency of a few lines, a fraction of a frame [9]. On the receiving side, a jitter buffer absorbs variation in network delay, and the decoder may hold frames too; both should be sized to the network you actually have, not left at streaming defaults.

8. Display

A display receives the picture line by line, but may buffer a full frame for scaling, frame-rate conversion, overdrive or picture enhancement. It then draws the frame from top to bottom over one refresh period, so the centre of the screen is updated about half a frame after the top. Display lag depends strongly on the mode; "game" or "low latency" modes disable much of the processing. Because manufacturers rarely publish it, it should be measured, and tools such as DispLagBox show that this can be done simply and with high precision [5].

Stage	Line-based design	Frame-based design	What adds the most
Sensor and readout	Up to 1 frame (rolling readout)	About 1 frame or more	Long exposure, low frame rate
Camera processing	A few lines	1 frame or more	Temporal filters, frame-rate conversion
Link (SDI, HDMI)	Negligible	Negligible	Converters, scalers, frame synchronisers
Capture card	Lines (sub-frame)	1 frame or more	Full-frame buffering before transfer
Driver and app queues	0 frames if newest frame is used	1 frame per queued buffer	Deep queues, processing every frame in order
Processing and AI	Inference time	Inference time + copies + batching	Batching, format conversions, render path
Encode / network / decode	Lines with JPEG XS or uncompressed IP	Frames with B-frames and lookahead	Default encoder settings, large jitter buffers
Display	About ½ frame (scan-out)	1 to several frames	Picture processing, scaling, frame-rate conversion

Rules of thumb for each stage. Real values depend on the product and its settings: measure them.

Example latency budgets

The following budgets show how the stages add up for four example architectures at 1080p and 60 fps (16.7 ms per frame). They are worked examples built from the rules of thumb above, not measurements of particular products. You can reproduce and change every one of them in the calculator below.

Stage	A. Direct view, hardware overlay	B. PC capture + AI overlay	C. Low-latency stream	D. Stream with default encoder
Camera (sensor + processing)	16.7	16.7	16.7	16.7
Capture	0.3	16.7	16.7	16.7
Application queue	0.0	16.7	0.0	16.7
Processing / AI	0.0	15.0	0.0	0.0
Render / compose	0.0	16.7	0.0	0.0
Encoder (frame + B-frames + lookahead)	0.0	0.0	16.7	733.3
Network + jitter buffer	0.0	0.0	35.0	120.0
Decode	0.0	0.0	16.7	16.7
Display (buffer + scan-out)	25.0	25.0	25.0	25.0
Total	**42.0**	**106.7**	**126.7**	**945.0**

Example budgets at 60 fps. Values in milliseconds.

Three lessons stand out. First, the direct-view chain is dominated by the camera and the display, not by the capture hardware, which is why a line-based capture and overlay path is worth having. Second, in the PC-based chain the AI model itself (15 ms) is a minor part of the total; the frames spent waiting in queues and in the render path cost more. Third, the same streaming system can be more than seven times slower with default encoder settings than with low-latency ones, which is the single biggest gain available in most streamed systems.

Latency budget calculator

Enter your own values to see where the time goes. Stages expressed in frames are converted with the frame rate you choose. Start from one of the four examples, then replace the assumptions with your measurements.

> **Interactive calculator.** Try your own values online: https://www.enciris.com/news/glass-to-glass-latency-white-paper/#latency-budget-calculator

How to measure glass-to-glass latency

Estimates are useful for design, but only a measurement tells you what the system really does. There are three practical methods, from quick to precise.

Method 1: a stopwatch on screen and a phone camera

Point the camera at a screen showing a running millisecond timer, and place the system's monitor next

to it. Film both with a phone in slow-motion mode and compare the two timer values in each frame of the recording. It costs nothing, but the result is limited by the refresh of both screens and the frame rate of the phone: at 240 fps, each recorded frame is 4.2 ms long. It is good enough to find a 100 ms problem, not to tune a 10 ms one.

Method 2: an LED and a photodiode

A microcontroller switches an LED in front of the camera and records, with a light sensor on the display, when the corresponding brightness change appears on screen. Because both events are timed by the same clock, each measurement is independent and precise: Bachhuber and Steinbach reached a precision of 0.5 ms with a 2 kHz sampling rate, using inexpensive parts [1]. Repeating the test thousands of times gives the full distribution, not just an average.

Method 3: timestamps inside the pipeline

Software timestamps at each stage (capture complete, inference complete, frame submitted for display) show where the time goes between the ends. They cannot see the sensor or the display, so they complement an optical measurement rather than replace it. Glass-to-algorithm latency can be measured the same way as G2G, with the light sensor replaced by the algorithm detecting the LED.

A measurement protocol that gives comparable results

1. Fix the video format, frame rate, exposure and every processing setting, and write them down with the result.
2. Put the light sensor or the timer in the centre of the screen, or measure top, centre and bottom: scan-out makes them differ by up to one frame.
3. Take at least a few hundred samples and report the median, the 95th percentile and the maximum, not only the mean. Latency varies from frame to frame because the event and the frame timing are not synchronised.
4. Measure the parts as well as the whole: camera to capture card, capture card to display, and the display alone with a known source.
5. Repeat after every change of firmware, driver, encoder setting or display mode. Latency regressions are rarely announced.
6. For medical and safety-related devices, keep the measurement setup and records with the design documentation, so the figure can be verified again later.

Design checklist: twelve ways to reduce latency

1. **Set a budget first.** Decide the acceptable end-to-end figure and give each stage a share, the way you would for power or cost.
2. **Raise the frame rate** when the light allows it: every frame of buffering anywhere in the chain gets shorter.
3. **Prefer line-based processing** in the camera, capture card and overlay path; avoid stages that need a complete frame.
4. **Keep the direct view direct.** Give the surgeon or operator a hardware loop-through output, and

run recording, streaming and AI in parallel rather than in series.
5. **Avoid converters and scalers** in the path, or check that they are line-based.
6. **Keep queues short** and always process the newest frame; drop late frames instead of catching up.
7. **Avoid copies** between capture, CPU and GPU memory, and convert colour formats once.
8. **Do not batch** real-time inference; measure the time from frame arrival to result, not only the model.
9. **Configure encoders for low latency:** no B-frames, no lookahead, small rate-control buffers, intra refresh instead of large key frames.
10. **Size jitter buffers** to the measured jitter of the real network, and prefer managed networks for live video.
11. **Choose displays in low-latency mode** and turn off picture enhancement on monitors used for live work.
12. **Measure end to end** with an optical method, and repeat after every change.

Choosing an interface: SDI, HDMI or IP

The interface rarely decides latency on its own, but it decides which stages you need. A comparison for real-time systems:

	SDI	HDMI	IP, uncompressed or JPEG XS	IP, H.264 / H.265
Added delay of the link itself	Negligible	Negligible	Lines to a few frames (packetisation, buffers)	Frames to seconds, depending on settings
Bandwidth	Up to 11.88 Gbit/s per 12G-SDI link	Up to 48 Gbit/s (HDMI 2.1), 96 Gbit/s (HDMI 2.2)	Network-dependent	Low (compressed)
Typical use	Medical, broadcast, industrial cameras	Monitors, consumer and prosumer sources	Broadcast and studio IP facilities	Remote viewing, recording, telemedicine
Standards	SMPTE ST 259, 292, 424, 2081, 2082	HDMI specification	SMPTE ST 2110, ISO/IEC 21122	ITU-T H.264 / H.265
Watch out for	Cable reach at high data rates	Locking connectors, cable length	Network design and timing	B-frames, lookahead, jitter buffers

Interfaces for live, low-latency video.

A common and robust pattern for medical and industrial systems is to keep SDI or HDMI for the live path to the operator, and to add an IP or compressed path in parallel for recording, remote viewing and analytics. The live path then never waits for the network.

How Enciris capture hardware fits in

Enciris designs capture cards, cameras and edge AI boards for medical, industrial and defense OEMs

with this budget in mind. The Lautrec LT-311, for example, captures four 12G-SDI inputs in Ultra HD 4K at up to 60 fps with less than one frame of latency, and has a direct HDMI 4K output with overlay [14]. On the 4K products, the combined loop-through and overlay delay is less than 20 lines of video [13], so an AI result can be drawn on the live picture without adding a frame to the surgeon's or operator's view.

For software integration, the LT-300 series is driven by a media server with a REST API and client libraries in Python, Go, C++ and C#, and the CH-007 carrier board brings NVIDIA Jetson Orin edge AI next to the capture card. Together they let you build the parallel architecture recommended above: a direct, line-based view for people, and a short, measured path to the algorithm.

Compare Enciris capture cards · Developer resources · Talk to an engineer about your latency budget

Frequently asked questions

What is glass-to-glass latency?

Glass-to-glass latency is the time from light entering the camera lens to the corresponding image leaving the display glass. It includes every stage in between: sensor readout, camera processing, the video link, capture, software, compression, the network and the display.

What is a good glass-to-glass latency for surgical video?

For telesurgery, a simulator study found 200 ms or less ideal and 300 ms suitable, and a 2022 study found delays of 100 ms or less acceptable. For local surgical video the target is tighter: keep the whole chain within a few frames, and the capture and overlay path within a frame or less.

How do you measure glass-to-glass latency?

The most reliable method uses an LED in front of the camera and a light sensor on the display, timed by the same microcontroller; it reaches sub-millisecond precision. A quicker method films a millisecond timer and the system display together with a slow-motion phone camera. Report the median, 95th percentile and maximum over many samples.

Does SDI have lower latency than HDMI?

Neither link adds meaningful delay by itself: both carry uncompressed video as it is produced. The difference comes from the devices around them, such as converters, scalers and displays that buffer frames. SDI is common in medical and industrial systems for its long cable runs and locking connectors.

Does H.264 or H.265 compression add latency?

It can add a lot. B-frames reorder frames and rate-control lookahead waits for future frames: x264 defaults to 3 B-frames and a 40-frame lookahead, which is 667 ms at 60 fps for the lookahead alone. Low-latency settings remove both. Codecs such as JPEG XS are designed for a latency of a few lines.

What does sub-frame latency mean?

Sub-frame latency means a stage passes the picture on before the complete frame has arrived, working on lines instead of whole frames. Its delay is therefore less than one frame time, for example less than 16.7 ms at 60 fps, and line-based stages can add well under a millisecond.

What is the difference between glass-to-glass and glass-to-algorithm latency?

Glass-to-glass latency ends when the image leaves the display and is what a person sees. Glass-to-algorithm latency ends when the image is available to a processing algorithm, such as an AI model, and is what limits machine reaction time. A system can be optimised for one without the other.

How much latency does a monitor add?

It depends on the monitor and its mode. Every display needs time to scan the picture out, about half a frame to the centre of the screen, and many add one or more frames of processing. Low-latency or game modes reduce this. Because manufacturers rarely publish it, measure it with a light sensor.

Conclusion

Glass-to-glass latency is the sum of many small decisions. The largest gains rarely come from a faster AI model or a faster network; they come from removing frames of buffering that nobody chose on purpose: in capture cards that wait for a full frame, in queues that fill up, in encoders left at their defaults and in displays processing the picture. Set a budget, design for lines rather than frames, keep the operator's view direct, and measure the result end to end.

Cite this white paper

```
Enciris Technologies, "Glass-to-Glass Latency: How to Measure and Reduce Video Delay in Medical and Industrial Systems," white paper, Version 1.0, November 2026. https://www.enciris.com/news/glass-to-glass-latency-white-paper/
```

References

1. C. Bachhuber and E. Steinbach, "A system for high precision glass-to-glass delay measurements in video communication," *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 2132–2136, 2016. doi:10.1109/ICIP.2016.7532735
2. C. Bachhuber, E. Steinbach, M. Freundl and M. Reisslein, "On the minimization of glass-to-glass and glass-to-algorithm delay in video communication," *IEEE Transactions on Multimedia*, vol. 20, no. 1, pp. 238–252, 2018. doi:10.1109/TMM.2017.2726189
3. S. Xu, M. Perez, K. Yang, C. Perrenot, J. Felblinger and J. Hubert, "Determination of the latency effects on surgical performance and the acceptable latency levels in telesurgery using the dV-Trainer simulator," *Surgical Endoscopy*, vol. 28, no. 9, pp. 2569–2576, 2014. doi:10.1007/s00464-014-3504-z
4. A. Nankaku, M. Tokunaga, H. Yonezawa, T. Kanno, K. Kawashima, K. Hakamada, S. Hirano, E. Oki, M. Mori and Y. Kinugasa, "Maximum acceptable communication delay for the realization of telesurgery," *PLOS ONE*, vol. 17, no. 10, e0274328, 2022. doi:10.1371/journal.pone.0274328

5. P. Stadler, A. Schmid and R. Wimmer, "DispLagBox: simple and replicable high-precision measurements of display latency," *Proceedings of Mensch und Computer 2020*, pp. 105–108, 2020. doi:10.1145/3404983.3410015
6. "Serial digital interface" (SMPTE ST 259, ST 292, ST 424, ST 2081, ST 2082, ST 2083 data rates), Wikipedia. https://en.wikipedia.org/wiki/Serial_digital_interface
7. HDMI Licensing Administrator, "Ultra96 HDMI Cable and HDMI 2.2 Specification." https://www.hdmi.org/spec21sub/ultra96
8. SMPTE, "SMPTE ST 2110: Professional Media Over Managed IP Networks" (standards suite). https://www.smpte.org/standards/ST2110
9. JPEG Committee, "Overview of JPEG XS" (ISO/IEC 21122). https://jpeg.org/jpegxs/
10. x264 project, `common/base.c`: default parameters and the "zerolatency" tune. https://code.videolan.org/videolan/x264/-/blob/master/common/base.c
11. FFmpeg Codecs Documentation, section "NVENC encoders" (`zerolatency` option). https://ffmpeg.org/ffmpeg-codecs.html
12. Linux kernel documentation, "Streaming I/O (Memory Mapping)," Video for Linux API. https://docs.kernel.org/userspace-api/media/v4l/mmap.html
13. Enciris, "Enciris 4K Latency-Free Overlay Features Explained." https://www.enciris.com/news/enciris-4k-latency-free-overlay-features-explained/
14. Enciris, "New Enciris Lautrec LT-311 brings Ultra HD 4K 60fps video capture with sub frame latency and real-time HDMI overlay." https://www.enciris.com/news/new-enciris-lautrec-lt-311-brings-ultra-hd-4k-60fps-video-capture-with-sub-frame-latency-and-real-time-hdmi-overlay/