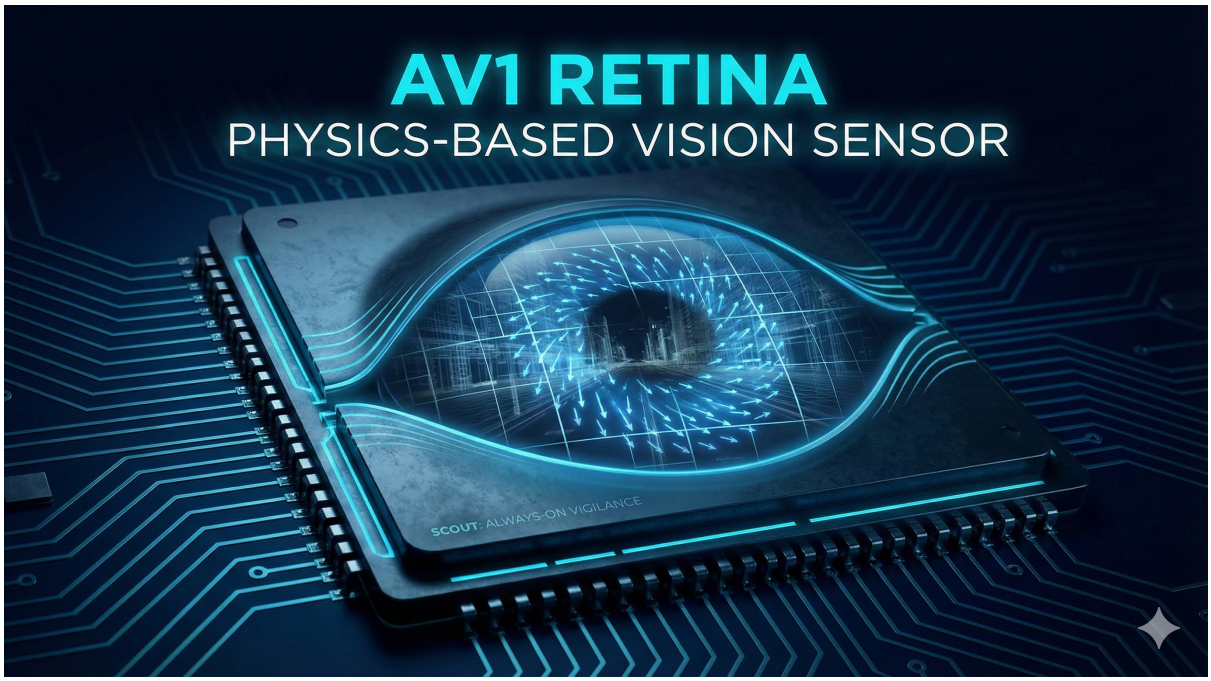
The AV1 Retina: Repurposing Video Codec Hardware as Low-Power Sensory Preprocessors for Machine Vision

Julien Pierre Salomon



Abstract

Modern computer vision systems suffer from a fundamental architectural inefficiency: they treat the camera as a passive data pipe, streaming massive quantities of raw pixels to power-hungry neural networks that must rediscover basic physics—motion, edges, depth—frame after frame. This paper proposes the **AV1 Retina**, a paradigm that repurposes commodity video encoder hardware as fixed-function sensory preprocessors. By intercepting the intermediate states of the AV1 encoding pipeline—Motion Vectors, Block Partitions, and Directional Filters—we extract "physics data" at milliwatt power scales, enabling a **Scout & Sniper** architecture that decouples physical event detection from semantic identification.

We validate this approach through analysis of three market-defining silicon platforms: **Google's Tensor G5** (the ideal edge "Scout"), **Qualcomm's Snapdragon 8 Elite** (a strategic gap), and **NVIDIA's Blackwell** (the centralized "Sniper" backend). Our findings demonstrate a $\sim$**30**$\times$ **power reduction** over conventional always-inference loops, reflex latencies bounded by sensor frame time ($\sim$17ms at 60fps), and a clear path to privacy-preserving machine perception using standard consumer silicon.

1 Introduction: The Raw Data Bottleneck

1.1 The Problem with Pixel-First Vision

The dominant paradigm in modern computer vision is **"Pixel-First."** In this model, the camera sensor acts as a passive data pipe, transmitting massive quantities of raw, uncompressed pixel data (RGB or YUV) to a central processor. Whether the task is detecting a pedestrian, stabilizing a drone, or monitoring a warehouse, the system must recalculate spatial and temporal relationships frame by frame, pixel by pixel.

This architecture creates what we term the **"Raw Data Bottleneck."** A standard 4K video stream at 60 frames per second generates approximately **12 gigabits of data per second**. Processing this flood of pixels to extract simple semantic truths—"something is moving on the left" or "an object is approaching"—requires activating the entire compute stack:

- **Image Signal Processor (ISP):** Demosaicing, white balance, tone mapping
- **DRAM Interface:** Buffering full-resolution frames
- **GPU or Neural Processing Unit (NPU):** Running inference on heavyweight models

An "always-inference" loop running a lightweight model like YOLOv8-nano on 4K input consumes **3.5 to 6 Watts** of continuous power. For battery-operated robotics, autonomous drones, and always-on security sensors, this power penalty is prohibitive.

Consider the mathematics: A drone with a 100Wh battery running continuous 4K inference at 5W has an operational window of **20 hours** if it does nothing but hover and watch. But drones also need power for flight motors (typically 200-400W), leaving only a fraction for perception. The result: battery life measured in **minutes**, not hours, with most energy spent re-discovering what the previous frame already knew.

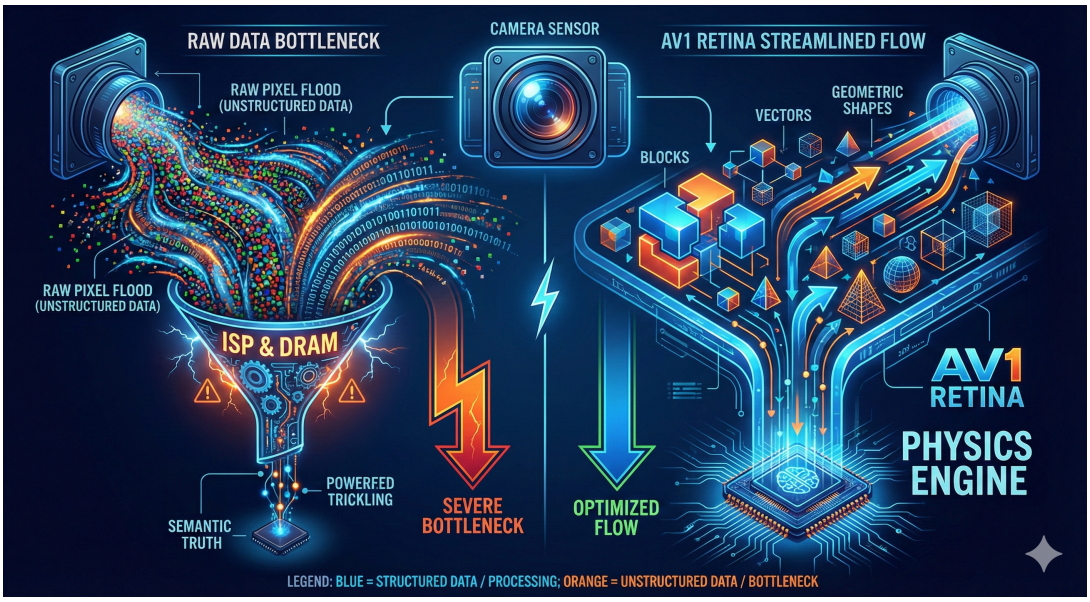


Figure 1: Visualizing the Raw Data Bottleneck vs. the AV1 Retina approach.

1.2 The Insight: Video Codecs Are Physics Engines

Here is the key observation that motivates the AV1 Retina: **video compression is physics computation**.

To compress video efficiently, an encoder must understand the physical structure of the scene. It cannot simply store pixels; it must:

1. **Calculate optical flow** (Motion Vectors) — understanding how objects move between frames
2. **Identify regions of complexity** (Block Partitioning) — distinguishing textured areas from flat backgrounds
3. **Separate signal from noise** (Quantization and Filtering) — preserving important details while discarding redundancy
4. **Detect edge directions** (CDEF) — enhancing perceptually important boundaries

These are precisely the operations that computer vision systems pay **billions of floating-point operations** to rediscover. A typical object detection network (YOLO, SSD, Faster R-CNN) includes:

- Convolutional layers that learn edge detectors
- Feature pyramid networks that identify multi-scale structure
- Region proposal networks that find areas of interest
- Temporal models (in video) that track motion

The video codec has already computed analogues of all these representations—but in **fixed-function silicon** that consumes milliwatts instead of watts.

1.3 The AV1 Retina Proposal

We propose treating the AV1 video encoder not as a storage tool, but as the **primary layer of a machine perception stack**. By intercepting the encoder's internal state before it completes the bitstream, we gain access to a rich set of "physics data":

Encoder Output	Physics Meaning	Vision Utility
Motion Vectors	Optical flow, velocity, trajectory	Object tracking, collision prediction, time-to-contact
Block Partition Tree	Structural complexity, texture density	Obstacle detection, landing zone identification, salie
CDEF Filter State	Edge direction, contrast	Lane keeping, segmentation preprocessing, bounda
Quantization Map	Signal-to-noise ratio, importance	Adaptive quality, low-light awareness, attention cueir
Reference Frame Index	Temporal consistency	Object persistence, occlusion handling
SAD (Matching Cost)	Prediction confidence	Scene change detection, anomaly flagging

Table 1: Mapping AV1 Encoder Outputs to Vision Utilities

A Video Processing Unit (VPU) operating in "Motion Estimation Only" mode—skipping entropy coding, reconstruction, and bitstream generation—can consume as little as **140 milliwatts**. This represents a $\sim$**30**$\times$ **reduction** in power compared to GPU inference.

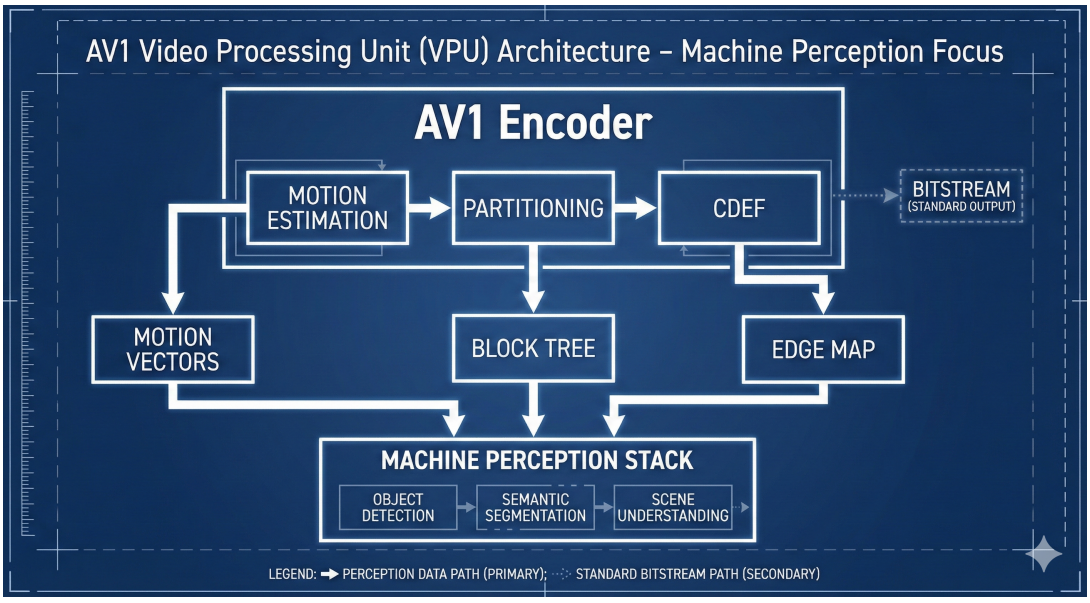


Figure 2: The AV1 Retina Proposal: Intercepting internal encoder states.

1.4 Why AV1 Specifically?

The AV1 codec, developed by the Alliance for Open Media (a consortium including Google, Netflix, Intel, NVIDIA, Apple, Amazon, and Microsoft), represents the state of the art in video compression. Several features make it particularly suited for machine vision:

Feature	AV1	H.264	HEVC
Max Superblock Size	128×128	16×16	64×64
Partition Types	Square + T-shaped	Square only	Square + asymmetric
Reference Frames	7	4	4
CDEF (Edge Filter)	✓Native	×	×
Warped Motion	✓(6-param affine)	×	×
Film Grain Synthesis	✓	×	×
Royalties	Free	MPEG-LA pool	HEVC Advance pool

Table 2: Comparison of AV1 features with H.264 and HEVC

The 128×128 superblock provides a **4$\times$ coarser grid** than HEVC's 64×64 for initial complexity assessment, while still supporting 4×4 granularity for fine detail. The native CDEF filter provides hardware edge detection. Warped Motion enables ego-motion compensation for moving cameras. And the royalty-free licensing enables deployment without legal complexity.

2 Biological Foundations: A Polymorphic Approach to Vision

In biological systems, the eye is not a camera—it is a specialized computer tuned to the survival needs of the species. A frog's eye "filters out" everything except small, moving objects (flies). A dog's eye sacrifices color and acuity for extreme motion sensitivity and high flicker fusion rates. A cat's eye amplifies light at the cost of resolution.

The architecture of the AV1 codec mirrors these functional specializations. Rather than mimicking human vision (a common but flawed approach in robotics, since humans are optimized for social cognition, not collision avoidance), we map codec primitives to the diverse visual systems evolution has optimized over hundreds of millions of years.

2.1 Motion Vectors $\rightarrow$ The Canine/Predator Paradigm

Biological System:

Canine vision is optimized for the detection of movement over chromatic detail. Dogs possess:

- **High rod cell density:** Rods detect motion and work in low light; dogs have $\sim$3$\times$ more rods than humans relative to cones
- **High flicker fusion threshold:** 70–80Hz vs. $\sim$60Hz in humans, enabling detection of rapid movements that appear as blur to us
- **Wide visual field:** $\sim$250° (vs. 180° in humans), providing peripheral motion awareness
- ***Tapetum lucidum*:** Reflective layer that amplifies available light, enabling dawn/dusk hunting

The result: a dog can spot a running squirrel in peripheral vision—a small, fast-moving object against a complex background—while a human fovea would register only a blur.

AV1 Mapping:

The Motion Estimation (ME) engine generates Motion Vectors (MVs) describing pixel displacement between frames. These vectors represent optical flow—the same data a predator's visual system extracts to lock onto prey.

Machine Vision Utility:

A system reading MVs can perceive velocity and trajectory instantly. It can "lock on" to a target defined solely by its motion signature—a non-zero vector cluster against a zero-vector background—without ever resolving the object's pixels.

Consider a security camera: Instead of running face detection on every frame, the Scout monitors the MV field. A human walking triggers a coherent, slow-moving vector cluster. A thrown object triggers a

Biological Feature	AV1 Equivalent	Function
Rod cell array	ME search window	Wide-area motion detection
High temporal resolution	Sub-pixel MV precision (1/8th pel)	Smooth trajectory estimation
Peripheral vision	Full-frame MV field	Global motion awareness
Target lock-on	MV cluster tracking	Object tracking without identity

Table 3: Mapping Canine Vision to AV1 Motion Estimation

fast, ballistic trajectory. A waving tree triggers random, incoherent vectors. The physics alone distinguish threat levels.

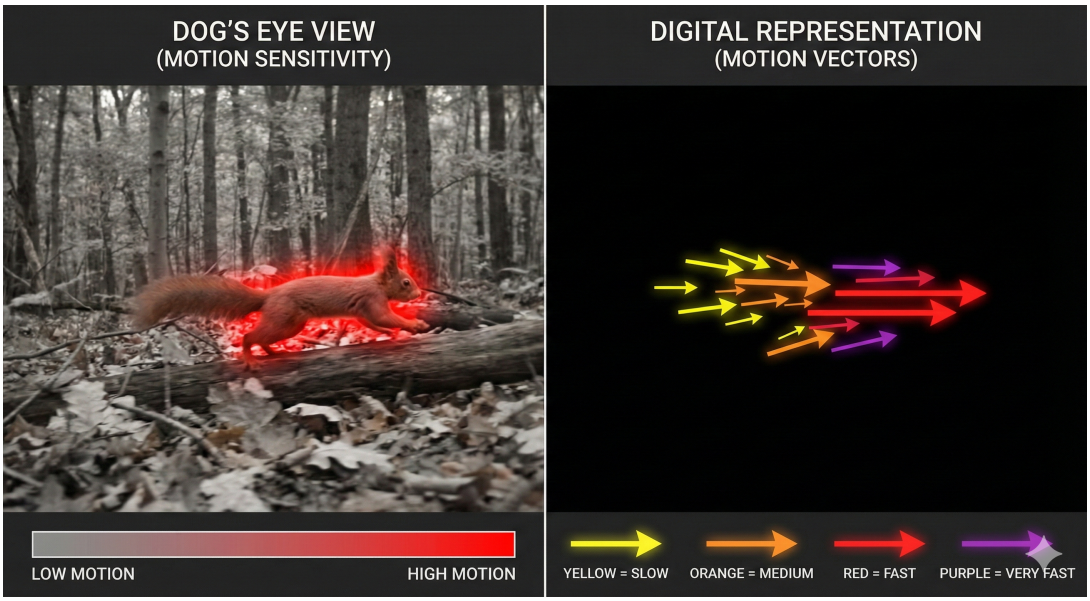


Figure 3: Biological Motion Detection vs. AV1 Motion Vectors.

2.2 Block Partitioning $\rightarrow$ The Insectoid/Compound Paradigm

Biological System:

Insect compound eyes (flies, dragonflies, bees) process the visual field as a mosaic. Each ommatidium (individual optical unit) covers $\sim 1^\circ$ of visual angle. The insect does not "focus" in the human sense—instead, it detects:

- **Variance:** Change in light intensity across the array
- **Disruption:** Patterns that violate expected optical flow
- **Expansion:** Looming stimuli (approaching predator or obstacle)

A fly doesn't see "a hand"—it sees "rapidly expanding threat pattern across ommatidia 47-89." This is why swatting a fly is hard: it reacts to the **physics of your motion**, not your identity.

AV1 Mapping:

AV1's 128×128 superblocks are recursively split based on image complexity. The encoder's Rate-Distortion Optimization (RDO) logic evaluates variance within each block:

- **Low variance** (flat area): Keep as large block $\rightarrow$ low entropy
- **High variance** (textured area): Split into smaller blocks $\rightarrow$ high entropy

The resulting **Partition Tree** is a dynamic saliency map, computed entirely in hardware.

Machine Vision Utility:

Biological Feature	AV1 Equivalent	Function
Ommatidia array	128×128 superblock grid	Spatial sampling
Variance detection	RDO split decision	Complexity measurement
Multi-resolution	Recursive partition ($128 \rightarrow 4$)	Adaptive detail
Looming detection	Expanding small-block region	Threat detection

Table 4: Mapping Insect Vision to AV1 Block Partitioning

The partition tree acts as a dynamic compound eye:

- **Large blocks** (128×128, 64×64): Navigable space (sky, road, wall)
- **Small blocks** (8×8, 4×4): Complex structure (obstacles, objects of interest)
- **Fragmentation over time**: A region that was large blocks and becomes small blocks is **changing**—something is happening there

A drone can identify landing zones (flat = large blocks) or obstacles (textured = small blocks) without running segmentation networks. The codec's RDO has already solved this.

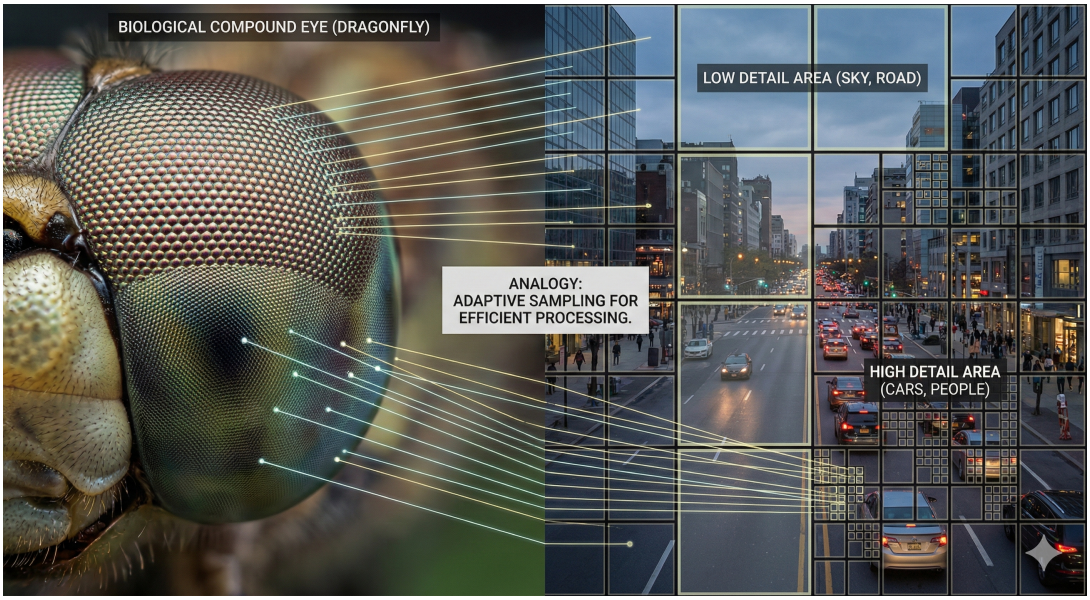


Figure 4: Insect Compound Eye vs. AV1 Block Partitioning.

2.3 Quantization $\rightarrow$ The Feline/Nocturnal Paradigm

Biological System:

Cats possess visual adaptations for nocturnal hunting:

- ***Tapetum lucidum*:** Reflective layer behind retina; light passes through photoreceptors twice, amplifying signal by $\sim$40%
- **Large cornea and pupil:** 6$\times$ larger than humans, gathering more light
- **Rod-dominated retina:** High sensitivity, low color discrimination
- **Reduced acuity:** $\sim$20/100 to 20/200 vision (legally blind by human standards)

The trade-off is explicit: **sacrifice color and detail for contrast and motion in low light**. A cat doesn't need to read the mouse's facial expression—it needs to detect the movement against a dark background.

AV1 Mapping:

The Quantization Parameter (QP) controls signal fidelity:

- **Low QP** (e.g., 10-20): High bitrate, fine detail preserved
- **High QP** (e.g., 40-50): Low bitrate, only high-contrast features survive

AV1 also supports **segment-based quantization**, allowing different QP for different regions within the same frame.

Biological Feature	AV1 Equivalent	Function
Light amplification	Low-bitrate mode	Operate on minimal signal
Rod dominance	High QP (discard color/texture)	Focus on edges/motion
Contrast sensitivity	Preserved high-frequency coefficients	Edge detection
Reduced acuity	Coarse quantization	Lower resolution, lower power

Table 5: Mapping Feline Vision to AV1 Quantization

Machine Vision Utility:

In low-power or low-light modes, the Scout system raises QP:

- Stops resolving fine texture and color
- Focuses entirely on high-contrast edges and motion
- Maintains awareness with a fraction of the power

This is hardware ”night vision”—the system remains alert to movement and structure while consuming milliwatts, trading chromatic and textural detail that a motion-detection system doesn't need.

2.4 CDEF $\rightarrow$ The Foveal Paradigm

Biological System:

The human fovea is a specialized region of the retina responsible for high-acuity central vision. It employs **lateral inhibition**—neighboring photoreceptors inhibit each other—to sharpen edges and enhance contrast at the point of focus. This is why we see sharp boundaries: the neural circuitry actively enhances them.

AV1 Mapping:

The Constrained Directional Enhancement Filter (CDEF) is an in-loop filter unique to AV1. For each 8×8 block, it:

1. Detects the dominant edge direction (8 candidates)
2. Applies smoothing **along** the edge (reduce noise)
3. Preserves sharpness **across** the edge (maintain boundary)

This is mathematically equivalent to lateral inhibition—enhancing perceptually important discontinuities while suppressing noise.

Biological Feature	AV1 Equivalent	Function
Lateral inhibition	Directional filtering	Edge enhancement
Foveal sharpening	Cross-edge preservation	Boundary detection
Noise suppression	Along-edge smoothing	Signal cleaning
Direction selectivity	8-way direction search	Orientation detection

Table 6: Mapping Foveal Vision to AV1 CDEF

Machine Vision Utility:

CDEF output provides a pre-conditioned edge map—hardware-accelerated edge detection that can feed lane-keeping, segmentation, or object detection without GPU cycles. The encoder has already found the edges; we just need to read them.

3 The Scout & Sniper Architecture

To operationalize the AV1 Retina, we propose a heterogeneous dual-sensor architecture. This design acknowledges a fundamental truth: **the optical requirements for detection (physics) and identification (semantics) are physically contradictory**.

A motion-detection sensor needs:

- Wide field of view (peripheral awareness)
- High temporal resolution (fast motion tracking)
- Monochrome sensitivity (no Bayer filter losses)
- Global shutter (accurate motion vectors)

An identification sensor needs:

- Narrow field of view (detail on target)
- High spatial resolution (facial features, text)
- Color sensitivity (semantic cues)
- High dynamic range (varied lighting)

No single sensor optimally satisfies both. The biological solution—separate retinal regions (fovea vs. periphery) with different properties—inspires our architecture.

3.1 Sensor A: The Scout (The Retina)

The Scout feeds the AV1 VPU. It is designed for purely physical observation—motion, structure, edges—without semantic content.

Specification	Value	Rationale
Resolution	VGA (640×480) to 720p	Sufficient for motion detection; frame fits in on-chip SRAM ($\sim$460KB for VGA YUV420); minimizes bandwidth and power
Color	Monochrome	No Bayer filter $\rightarrow \sim 3\times$ light sensitivity; full pixel utilization; motion doesn't need color
Shutter	Global	All pixels exposed simultaneously; accurate motion vectors free of rolling-shutter skew artifacts
Frame Rate	60–120 fps	High temporal resolution for fast motion; matches predator flicker fusion rates
FOV	Ultra-wide (120°–160°)	Peripheral awareness; threat detection from any direction
Optics	Fixed focus / deep DOF	No autofocus latency; everything in focus for motion detection
Data Path	Sensor $\rightarrow$ VPU $\rightarrow$ MCU	No GPU involvement; pure hardware physics extraction

Table 7: Scout Sensor Specifications

Example Hardware:

- OmniVision OV7251 (VGA global shutter, $4 sensor)

- Sony IMX219 (1080p rolling, but adequate for cost-sensitive applications)
- onsemi AR0144 (1MP global shutter, industrial-grade)

Power Budget:

Component	Typical Power
VGA global shutter sensor	30–50 mW
MIPI CSI-2 interface	5–10 mW
Clock/control logic	5 mW
Total Scout Sensor	**$\sim$50 mW**

3.2 Sensor B: The Sniper (The Fovea)

The Sniper feeds the Neural Network. It is designed for semantic understanding—classification, identification, recognition.

Specification	Value	Rationale
Resolution	4K (3840$\times$2160) to 8K	Detail for facial recognition, license plates, text OCR
Color	RGB (Bayer or stacked)	Semantic cues: clothing color, signage, skin tone
Shutter	Rolling	Acceptable for static/slow capture; enables larger pixels
Frame Rate	30 fps	Standard video rate; semantic analysis is slower anyway
FOV	Narrow (35mm–85mm equiv.)	Focus on region of interest identified by Scout
Optics	Autofocus / zoom	Adapt to target distance
Data Path	Sensor $\rightarrow$ ISP $\rightarrow$ NPU/GPU	Full inference pipeline

Table 8: Sniper Sensor Specifications

Example Hardware:

- Sony IMX989 (1-inch, 50MP, flagship smartphone sensor)
- Samsung ISOCELL HP2 (200MP, pixel binning to 12.5MP)
- OmniVision OV48C (48MP, mid-range)

Power Budget (When Active):

Component	Typical Power
4K RGB sensor	200–400 mW
ISP (debayer, tone map, AWB)	200–300 mW
DRAM (frame buffering)	400–600 mW
NPU/GPU (object detection)	1,500–5,000 mW
Total Sniper Active	**2.5–6.5 W**

Key Insight: The Sniper is power-gated during normal operation. It consumes **zero power** in State 0 (Deep Watch). The Scout's job is to minimize how often the Sniper wakes.

3.3 The Reflex Arc: Operational States

This architecture creates a biological "reflex arc"—a fast path that bypasses the slow cognitive brain. In mammals, a hand touching a hot stove triggers muscle contraction **before** the pain signal reaches the cortex. The Scout enables analogous machine reflexes.

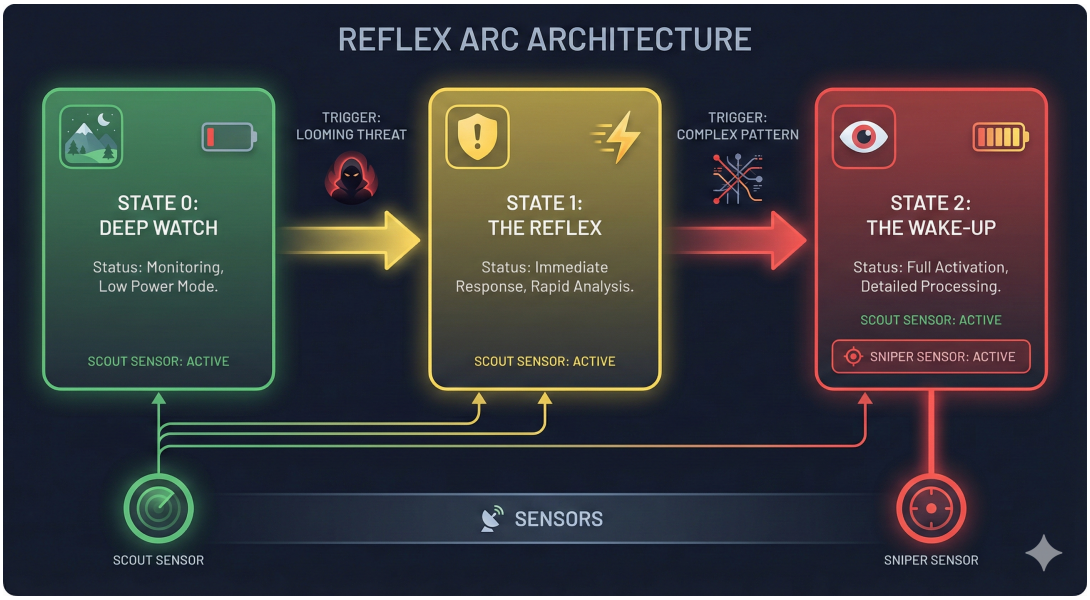


Figure 5: The Reflex Arc: Operational States of the AV1 Retina.

3.4 Trigger Conditions: What Wakes the Sniper?

The Scout monitors the Motion Vector field and Block Partition map for specific patterns that indicate semantic analysis is needed:

Trigger	MV/Partition Pattern	Response
Looming Threat	Radially expanding MV cluster (approaching object)	STATE 1: Immediate reflex
Intrusion	New MV cluster entering monitored zone	STATE 2: Wake Sniper for ID
Sustained Motion	Persistent MV activity $>$ N seconds	STATE 2: Check if human/animal/vehicle
Complex Pattern	High partition fragmentation + motion	STATE 2: Something unusual happening
Scene Change	Global SAD spike (new scene)	STATE 2: Re-establish baseline
Periodic Motion	Repetitive MV pattern	Suppress (tree swaying, fan, etc.)

Table 9: Trigger Conditions for Sniper Activation

Key Insight: The triggers are **physics-based**, not semantic. We're not asking "Is that a person?"—we're asking "Is something approaching fast?" or "Did something new appear?" The physics questions are cheap; the semantic questions are expensive.

3.5 Power Analysis: The Quantitative Case

The justification for this architectural complexity lies in the **cost per query**. In a standard vision system, asking "Is there a threat?" costs the same energy as asking "Is that threat a cat or a dog?" The AV1 Retina decouples these costs.

3.5.1 Scenario A: Standard Always-Inference Loop

Battery Life (100Wh): $\sim$22 hours (perception only, no actuation)

3.5.2 Scenario B: AV1 Retina Watchdog Loop

Battery Life (100Wh): $\sim$690 hours ($\sim$29 days)

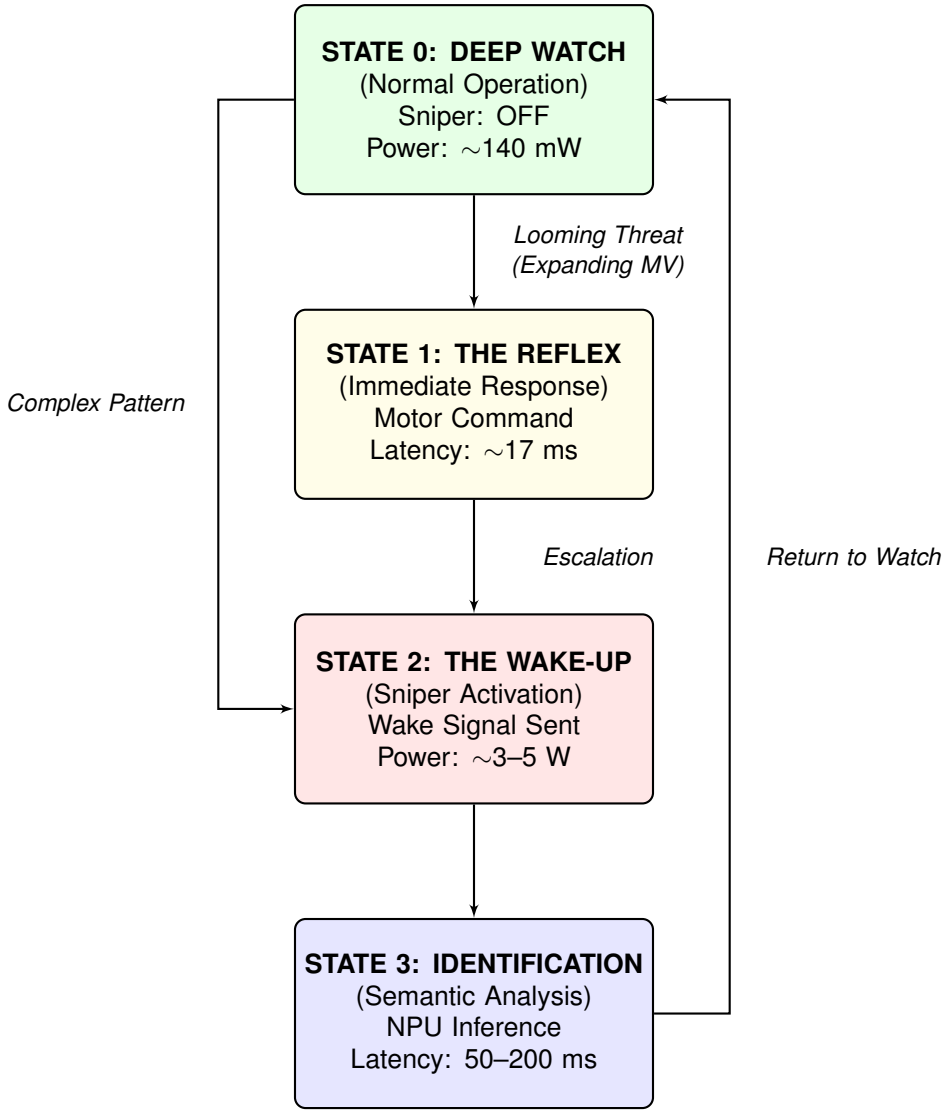


Figure 6: The Reflex Arc: Operational States of the AV1 Retina.

3.5.3 The Power Ratio

$$\text{Power Reduction} = \frac{4550 \text{ mW}}{145 \text{ mW}} = 31\times$$

In realistic scenarios where the Sniper wakes more frequently (e.g., 10×/hour in a busy environment), the effective power rises to ∼190 mW, still achieving a **24× reduction**.

Result: A device can maintain visual vigilance for **weeks** on a small battery, consuming high-wattage inference only when physics demand it.

4 AV1 as a Machine Vision Codec: Technical Mapping

The AV1 codec specification, developed by the Alliance for Open Media (Google, Netflix, Intel, NVIDIA, Apple, and others), represents the state of the art in video compression. More importantly for our purposes, its sophisticated prediction tools create rich intermediate representations that map directly to machine vision primitives.

Component	Power	Duty Cycle	Effective Power
Sensor (4K, 30fps)	300 mW	100%	300 mW
ISP (Debayer, Tone Map)	250 mW	100%	250 mW
DRAM (Frame Buffering)	500 mW	100%	500 mW
GPU/NPU (Object Detection)	3,500 mW	100%	3,500 mW
Total Continuous Draw			**4,550 mW**

Component	Power	Duty Cycle	Effective Power
Scout Sensor (VGA, Mono)	50 mW	100%	50 mW
VPU (ME-Only Mode)	80 mW	100%	80 mW
MCU (Threshold Check)	10 mW	100%	10 mW
Sniper (Wakes 1×/hour)	5,000 mW	0.1%	5 mW
Total Effective Draw			**145 mW**

4.1 Motion Vectors: Optical Flow in Hardware

The Motion Estimation (ME) engine is the core of the ”Scout” sensor. It exploits temporal redundancy by searching for matching blocks in reference frames, outputting vectors that describe pixel displacement between frames.

4.1.1 Technical Properties

Property	AV1 Specification	Vision Utility
Precision	1/8th pixel (3 fractional bits)	Smooth trajectory estimation for slow-moving objects
Reference Frames	Up to 7 (LAST, LAST2, LAST3, GOLDEN, BWDREF, ALTREF2, ALTREF)	Track objects through occlusion; temporal consistency
Search Range	Implementation-dependent (typically ±512 pixels)	Detect fast motion; wide-area awareness
Block Sizes	Variable (128×128 down to 4×4)	Adaptive resolution based on motion complexity

4.1.2 Compound Prediction

AV1 can blend predictions from multiple reference frames using weighted averaging:

$$P_{compound} = w_1 \cdot R_1[x + mv_1] + w_2 \cdot R_2[x + mv_2]$$

This provides a richer motion model, particularly for partially occluded objects or alpha-blended regions.

4.1.3 Extracted Data Structure

For each block, the ME engine outputs:

SAD as Confidence Metric:

- **Low SAD** = excellent match = predictable motion = background or smoothly moving object
- **High SAD** = poor match = unpredictable = new object, occlusion boundary, or scene change

A Scout system can use SAD spikes to detect scene-change events or object boundaries without semantic analysis.

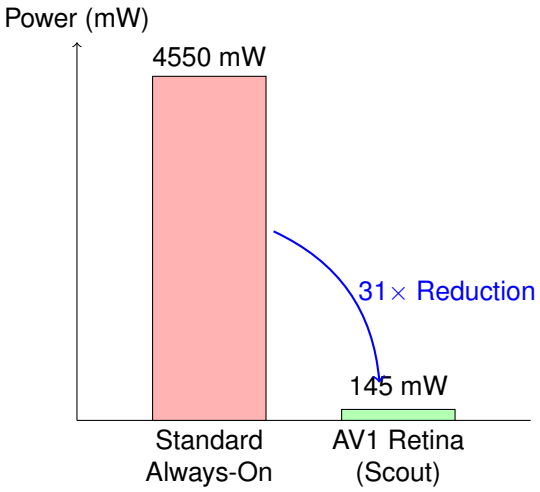


Figure 7: Power Consumption Comparison.

Field	Type	Meaning
`mv.x`	int16	Horizontal displacement ($\times$8 for sub-pixel)
`mv.y`	int16	Vertical displacement ($\times$8 for sub-pixel)
`ref_frame`	uint8	Which reference frame (0–6)
`sad`	uint32	Sum of Absolute Differences (match quality)
`mode`	enum	Prediction mode (NEWMV, NEARESTMV, etc.)

4.1.4 MV Reliability: When the Encoder "Lies"

A critical caveat for vision applications: **Motion Vectors are optimized for compression, not ground truth**. The encoder's Rate-Distortion Optimizer may choose a "wrong" MV if it reduces bitrate. For example:

- A textured region sliding behind another textured region may generate an MV pointing to the *background* if the texture matches better
- Repeating patterns (fences, bricks, waves) can produce aliased MVs pointing to wrong periods
- The encoder may use NEARESTMV or NEARMV prediction modes that reuse neighbor vectors even when the true motion differs

Mitigation: SAD as Confidence Metric

The Sum of Absolute Differences provides a built-in confidence signal:

SAD Value	Interpretation	Action
SAD $<$ threshold$_1$	High confidence—excellent match	Trust MV fully
threshold$_1$ $<$ SAD $<$ threshold$_2$	Medium confidence—usable	Apply IIR temporal filter
SAD $>$ threshold$_2$	Low confidence—poor match	Discard or flag for attention

Practical thresholds depend on block size and content, but a rule of thumb: SAD $> 20\times$ mean SAD indicates an unreliable vector. By filtering out high-SAD vectors, the Scout can maintain a robust optical flow field despite encoder optimization decisions.

4.1.5 Looming Detection: Time-to-Contact from MV Divergence

One of the most safety-critical capabilities enabled by Motion Vectors is **looming detection**—identifying objects approaching the camera on a collision course. This exploits a fundamental property of optical flow: approaching objects produce radially expanding vector fields.

The Physics:

For an object approaching at velocity v_z along the optical axis, the image expansion rate relates to time-to-contact τ (tau):

$$\tau = \frac{1}{\nabla \cdot \mathbf{v}} = \frac{1}{\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}}$$

Where $\nabla \cdot \mathbf{v}$ is the divergence of the motion vector field. This is derived from the geometric relationship:

$$\tau = \frac{Z}{v_z} = \frac{f}{\dot{s}/s}$$

Where Z is distance, v_z is approach velocity, f is focal length, and $\dot{s}/s$ is the relative size expansion rate.

Hardware Implementation:

The Scout computes divergence by differencing neighboring Motion Vectors:

```
div_x = MV[x+1, y].x - MV[x-1, y].x
div_y = MV[x, y+1].y - MV[x, y-1].y
divergence = (div_x + div_y) / 2
tau = 1.0 / divergence  // time-to-contact in frames
```

Divergence	Tau (frames @ 60fps)	Interpretation
> 0.5	< 2 (33 ms)	**CRITICAL: Imminent collision**
0.1 - 0.5	2-10 (33-166 ms)	Approaching object—alert
< 0.1	> 10 ($>$166 ms)	Slow approach or lateral motion
< 0	N/A	Receding object

Key Insight: Looming detection requires no object recognition. A drone doesn't need to know *what* is approaching—only that *something* is approaching *fast*. End-to-end reflex latency is bounded by sensor frame time ($\sim$17ms at 60fps), not compute.

4.2 Recursive Partitioning: The Saliency Map

AV1's superblock structure (128×128 pixels, recursively split to 4×4) creates a natural saliency representation that mirrors the insectoid compound eye.

4.2.1 The Partition Decision Tree

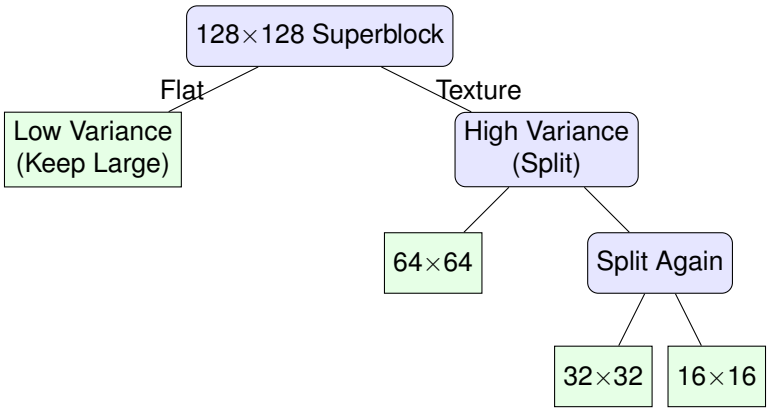


Figure 8: The Partition Decision Tree: Recursive splitting based on complexity.

4.2.2 Partition Depth as Complexity Signal

Partition Depth	Block Size	Typical Content	Vision Interpretation
0	128×128	Clear sky, blank wall	Navigable space; ignore
1	64×64	Gradual gradients	Low-priority region
2	32×32	Moderate texture	Potential object
3	16×16	High detail (faces, text)	Object of interest
4	8×8	Fine edges, complex texture	Edge/boundary region
5	4×4	Maximum complexity	Critical detail zone

Vision Utility:

- **Block Size = Inverse Complexity:** Large blocks indicate navigable space; small blocks indicate obstacles or objects of interest
- **Temporal Stability:** A region that remains large blocks across frames is static background; a region that fragments is undergoing change
- **Attention Cueing:** The partition map tells the Sniper exactly where to focus its high-resolution analysis

4.2.3 Temporal Partition Filtering: Stabilizing the Saliency Map

A challenge for vision applications: **partition trees are not temporally stable**. The encoder's RDO makes independent decisions per frame, so a block at the boundary between "split" and "don't split" may oscillate between depths across frames—even with no actual scene change.

The Problem:

Frame N: Block (5, 3) $\rightarrow$ Depth 2 (32×32)
Frame N+1: Block (5, 3) $\rightarrow$ Depth 3 (16×16) $\leftarrow$ RDO made different choice
Frame N+2: Block (5, 3) $\rightarrow$ Depth 2 (32×32) $\leftarrow$ Back again

This "partition flicker" would cause false triggers in a naive Scout system.

Solution: IIR Temporal Filtering

Apply an Infinite Impulse Response filter to the partition depth map:

$$D_{filtered}[t] = \alpha \cdot D_{raw}[t] + (1 - \alpha) \cdot D_{filtered}[t - 1]$$

Where:

- $D_{raw}[t]$ is the current frame's partition depth
- $D_{filtered}[t - 1]$ is the previous filtered depth
- $\alpha \in [0.1, 0.3]$ for slow adaptation (background stability)
- $\alpha \in [0.5, 0.8]$ for fast adaptation (quick object detection)

Implementation:

```
for each superblock (x, y):
    depth_filtered[x,y] = 0.2 * depth_raw[x,y] + 0.8 * depth_filtered_prev[x,y]
    if |depth_filtered[x,y] - depth_filtered_prev[x,y]| > threshold:
        trigger_attention(x, y)  // Genuine change, not noise
```

This low-pass filter costs negligible compute (one multiply-add per block) but dramatically reduces false positives from encoder RDO noise.

4.3 Warped Motion: Ego-Motion Compensation

AV1's Warped Motion tool predicts blocks using affine transformations (rotation, zoom, shear). This is critical for moving platforms and directly mirrors the biological Vestibulo-Ocular Reflex (VOR).

4.3.1 The Ego-Motion Problem

When the camera moves, everything in the scene generates motion vectors. A naive Scout monitoring for "motion" would trigger constantly during camera pan, tilt, or ego-motion.

4.3.2 The Affine Motion Model

AV1 computes global motion parameters that describe camera transformation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Where:

- α, δ: Scale factors (zoom)
- β, γ: Rotation/shear components
- t_x, t_y: Translation

4.3.3 Hardware Implementation

The encoder estimates global motion using the Optical Flow field, then:

1. Computes the affine transformation that minimizes residual error
2. Signals this transformation in the bitstream
3. For each block, generates a warped prediction from the reference frame

Vision Utility:

Motion Type	Global Motion	Local MV	Scout Interpretation
Static scene + camera pan	High (t_x, t_y)	$\sim$Zero	Normal—camera moving
Static scene + camera zoom	High α, δ	$\sim$Zero	Normal—camera zooming
Moving object + static camera	$\sim$Zero	Non-zero cluster	**ALERT: Independent object motion**
Moving object + camera pan	High global	Residual non-zero	**ALERT: Object moving differently than background**

A drone can distinguish between "I'm rotating left" (global flow) and "that bird is flying toward me" (local flow anomaly). This is hardware-computed VOR.

4.4 CDEF: Hardware Edge Detection

The Constrained Directional Enhancement Filter operates on 8×8 blocks, detecting the dominant edge direction among 8 candidates and applying directional smoothing. This is essentially a **hardware Canny edge detector**.

4.4.1 Direction Search

For each 8×8 block, the hardware evaluates 8 directional paths:

The hardware computes variance along each path and selects the direction with **minimum variance** (the edge runs along this direction).

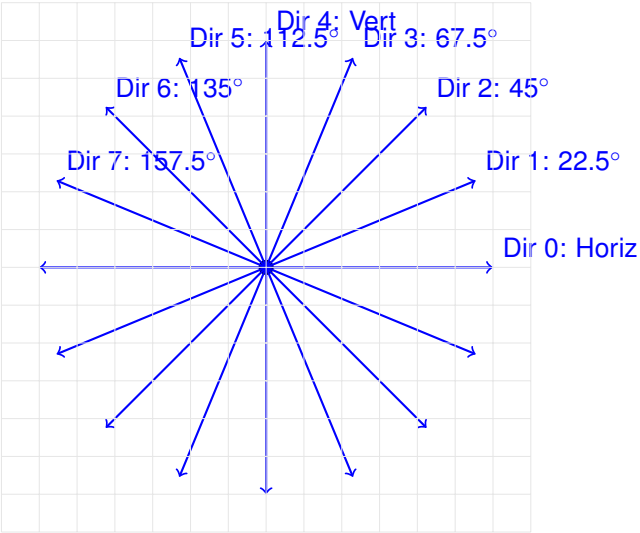


Figure 9: The 8 CDEF Directional Modes.

4.4.2 Directional Filtering

Once direction is identified:

- **Along the edge:** Apply smoothing (reduce noise)
- **Across the edge:** Preserve sharpness (maintain edge contrast)

This non-linear, direction-aware filtering is exactly what edge-aware smoothing algorithms (bilateral filter, guided filter) achieve—but implemented in fixed-function hardware.

4.4.3 Vision Utility

The CDEF output provides:

1. **Direction Map:** 8 possible edge orientations per 8×8 block
2. **Strength Map:** Filter strength indicates edge confidence
3. **Pre-filtered Image:** Noise suppressed, edges preserved

For lane-keeping, the direction map alone can identify road edges. For segmentation, the strength map highlights object boundaries. All computed for free during the encode.

4.5 Depth from Motion: Monocular 3D Without Stereo

Motion Vectors enable **monocular depth estimation** through motion parallax—closer objects produce larger MVs than distant objects when the camera translates.

4.5.1 The Motion Parallax Principle

For a camera translating with velocity (V_x, V_y, V_z), the optical flow at image point (x, y) is:

$$v_x = \frac{V_x \cdot f - V_z \cdot x}{Z}$$
$$v_y = \frac{V_y \cdot f - V_z \cdot y}{Z}$$

Where Z is scene depth and f is focal length. Inverting:

$$Z = \frac{V_x \cdot f - V_z \cdot x}{v_x}$$

Key Insight: Given known camera motion (from IMU or odometry), depth is inversely proportional to MV magnitude.

4.5.2 Practical Depth Estimation

For a forward-moving drone or robot with primarily V_z motion:

MV Magnitude	Relative Depth	Interpretation
Large MV ($>$ 10 px)	Close	Obstacle in path
Medium MV (3-10 px)	Mid-range	Objects of interest
Small MV ($<$ 3 px)	Far	Background/horizon
Zero MV	Infinity or stationary	Sky, distant terrain

Limitations:

- Requires camera translation (rotation produces global flow with no depth signal)
- Degenerate cases: objects moving toward camera at ego-speed produce zero MV
- Accuracy degrades with distance (small MVs, quantization noise)

Utility: Even coarse depth ($near$ / mid / far bins) is sufficient for obstacle avoidance. The Scout provides this for free; the Sniper can refine with stereo or neural depth when needed.

4.6 Quantization as Adaptive Sensitivity

The Quantization Parameter (QP) controls the trade-off between bitrate and quality. In the Scout context, it provides adaptive sensitivity control.

4.6.1 Segment-Based QP

AV1 allows different QP values for different **segments** of the frame. The encoder can:

- Use low QP (high detail) for regions of interest
- Use high QP (low detail) for background

Vision Utility: The QP map is effectively a **hardware-generated attention map**. Regions where the encoder allocated more bits are regions the rate-distortion optimizer deemed ”important.”

4.6.2 Low-Light Mode

In low-light or low-power scenarios:

- Raise QP globally
- Preserve only high-contrast edges and motion
- Sacrifice texture detail

This provides hardware ”night vision”—the Scout continues to detect motion and edges while consuming minimal power, trading chromatic and textural detail that a motion-detection system doesn’t need.

5 Hardware Micro-Architecture: How the Silicon Works

Understanding why hardware encoding is so efficient—and why it can serve as a vision preprocessor—requires examining the micro-architecture of modern VPUs at the gate level. The AV1 codec presents unique challenges that have driven innovations in silicon design, from breaking the serial bottlenecks of arithmetic decoding to managing the massive memory bandwidth of multi-reference motion compensation.

5.1 Fixed-Function Logic vs. Programmable Cores

The fundamental efficiency gain comes from eliminating the instruction overhead of general-purpose processors.

Operation	CPU/GPU Shader	Fixed-Function ASIC
Fetch instruction	✓(costs energy)	×(hardwired)
Decode instruction	✓(costs energy)	×(hardwired)
Configure ALU	✓(costs energy)	×(hardwired)
Execute operation	✓	✓

A CPU or GPU must pay an "energy tax" for every operation—fetching and decoding the instruction that tells it what to do. An ASIC has no instructions. When a pixel enters the CDEF unit, it flows through the logic gates automatically. The data path from input to output is hardwired. This eliminates the fetch/decode overhead, improving efficiency by **10× to 100×**.

5.2 The Entropy Decoding Front-End: Breaking the Serial Bottleneck

The entry point for any video decoder is the entropy decoding stage, where the compressed bitstream is parsed into syntax elements (prediction modes, motion vectors, transform coefficients). AV1 uses a Multi-Symbol Arithmetic Coder designed for higher throughput than HEVC's binary CABAC, but this introduces unique hardware challenges.

5.2.1 The Serial Dependency Problem

Arithmetic coding achieves compression approaching the Shannon limit by representing symbol sequences as sub-intervals within the range [0, 1). The core challenge for hardware is sequential dependency: **each symbol's decoding depends on the state (Range and Offset) left by the previous symbol**.

5.2.2 Pipelined Hardware Architecture

Modern AV1 hardware mitigates this bottleneck through deep pipelining, typically consisting of four distinct stages:

The separation of Context Selection (Stage 1) and Range Update (Stage 3) is crucial. By pre-fetching probabilities before the arithmetic engine needs them, the design hides SRAM latency that would otherwise stall the pipeline.

5.2.3 rLPS Forwarding: Speculative Execution in Hardware

The most significant innovation in high-throughput AV1 entropy decoders is **rLPS (Range Least Probability Symbol) Forwarding**. In a standard loop, the hardware must wait for the current range update to finish before starting the next.

With rLPS forwarding, the hardware speculates on both outcomes:

1. Calculate potential next Range for **Most Probable Symbol (MPS)**
2. Calculate potential next Range for **Least Probable Symbol (LPS)** in parallel

Pipeline Stage	Function	Hardware Activity
Stage 1: Context Selection	Determine probability model index	Logic analyzes previous syntax elements (neighbors) to select CDF address in SRAM
Stage 2: Probability Fetch	Retrieve the CDF from local memory	Access on-chip SRAM to fetch probability distribution for current symbol context
Stage 3: Range Update	Update interval width (Critical Path)	Multipliers/Shifters update Range variable based on symbol probability
Stage 4: Symbol Extraction	Output symbol and normalize	Comparisons determine symbol value; Offset shifted to maintain precision

3. Once current symbol is resolved, a multiplexer selects the correct pre-calculated range
4. Forward immediately to next stage

This removes the arithmetic logic delay from the critical feedback loop, allowing clock frequency to be limited only by multiplexer switching time rather than the full multiplier delay.

5.2.4 Multi-Symbol Processing

Unlike HEVC's strictly binary CABAC, AV1's engine can process non-binary alphabets (up to 16 symbols) for syntax elements like intra-prediction modes. While this increases comparison logic complexity (requiring parallel comparators), it increases bits decoded per clock cycle. Advanced implementations employ "super-scalar" parsing or dual-core entropy engines essential for 8K HDR content.

5.3 The Motion Estimation Engine: The Scout's Core

The Motion Estimation (ME) engine is the heart of the Scout sensor. It exploits temporal redundancy by searching for matching blocks in reference frames, outputting the Motion Vectors that form the basis of physics-first perception.

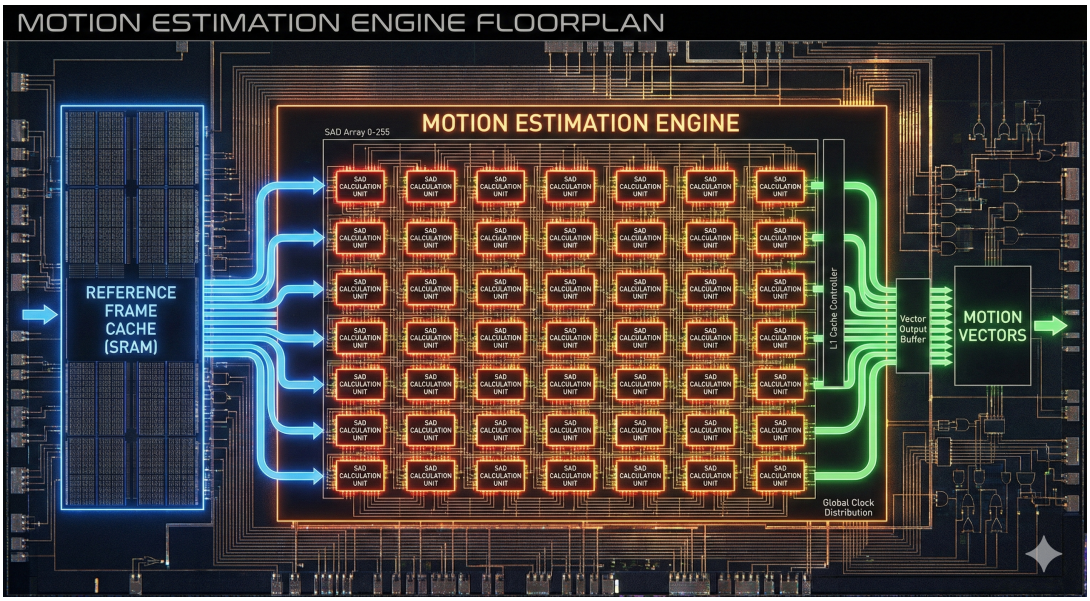


Figure 10: Hardware Architecture of the Motion Estimation Engine.

5.3.1 Hierarchical Search Architecture

Coarse-to-Fine Strategy:

1. **Level 0 (Coarse):** Search on 4× downscaled frame; find approximate match region
2. **Level 1 (Medium):** Refine on 2× downscaled frame within the candidate region
3. **Level 2 (Fine):** Full-resolution search within a small window around the refined candidate

This reduces the search space from potentially millions of candidate positions to hundreds, with minimal quality loss.

5.3.2 SAD Array: Parallel Block Matching

The Sum of Absolute Differences (SAD) computation is the core operation of motion estimation. Hardware implements this as a massively parallel array:

$$SAD = \Sigma|Current[x, y] - Reference[x + mvx, y + mvy]|$$

Hardware Implementation:

- Array of parallel subtractors (one per pixel in the block)
- Absolute value circuits (trivial in hardware: conditional negation)
- Adder tree to sum all differences in $\log_2$(N) stages
- For a 16×16 block: 256 parallel subtract-abs operations → 8-stage adder tree

Throughput: A well-designed SAD array can evaluate multiple candidate positions per clock cycle, enabling real-time search across large windows.

5.3.3 Motion Vector Prediction and Caching

In AV1, Motion Vectors are differentially encoded relative to predictors. The hardware extraction process involves:

1. **Candidate List Construction:** The MV Prediction Unit scans spatial neighbors (blocks to the left and above) and temporal neighbors (collocated blocks in reference frames) to build a candidate list. This requires rapid access to a **Motion Vector Cache**—specialized SRAM storing MV fields of recent frames.
2. **Delta Decoding:** The entropy decoder outputs a delta value. Hardware adds this to the selected candidate to recover the actual MV.
3. **High-Precision Refinement:** AV1 MVs support **1/8th pixel precision**. These are stored as fixed-point integers (14-bit values where the lower 3 bits represent the fractional part). This precision enables smooth trajectory estimation but requires wider data paths throughout the prediction logic.

5.3.4 Power Efficiency Analysis

Architecture	Resolution	Power	Process Node
Academic H.264 ME core	CIF (352×288)	2.13 mW	65nm
Industry reference ME	4K UHD 60fps	107.76 mW	28nm
Scaled VGA estimate	VGA (640×480)	∼4 mW	28nm
Modern 4nm projection	VGA (640×480)	<**2 mW**	4nm

The critical insight: The ME engine consumes milliwatts because it performs only integer additions and comparisons—no floating-point math, no weight fetches, no activation functions. The ”Scout” needs only this engine; the expensive entropy coding and reconstruction paths can be power-gated.

5.4 The Interpolation Filter Engine

Since motion vectors point to fractional pixel locations, the hardware must synthesize pixels that don't exist in the source data.

5.4.1 Separable 2D Filtering

To minimize logic complexity, 2D interpolation is decomposed into two 1D operations:

1. **Horizontal filter** generates intermediate values for an entire row
2. **Vertical filter** processes those intermediate results column-wise

5.4.2 8-Tap FIR Filter Implementation

AV1 uses high-quality 8-tap Finite Impulse Response filters. In hardware:

- For a 4×4 block: 4×9 horizontal operations (due to filter support window) + 4×4 vertical operations
- Each tap requires a multiplication and accumulation

Shift-and-Add Optimization: To reduce multiplier area, specific filter coefficients (fixed in the AV1 spec) are implemented using shift-add logic. For example:

- Multiplication by 10 $\rightarrow$ `(x << 3) + (x << 1)` (shift by 3, shift by 1, add)
- Uses simple wire shifts and a single adder rather than a complex multiplier circuit

5.5 The Intra-Prediction Engine

When blocks are encoded without temporal reference, the hardware relies on Intra-Prediction. AV1 expands directional modes to **56 angles** and introduces powerful non-directional predictors.

5.5.1 Directional Extrapolation Hardware

- **Line Buffers:** Engine reads pixels from a "neighbor buffer" containing the bottom row of the block above and right column of the block to the left
- **Angle Calculation:** For each of 56 nominal angles, a lookup table determines the step size (how much source coordinate shifts per row/column)
- **Sub-Pixel Interpolation:** Bilinear interpolation estimates values between integer boundary pixels for smooth gradients at oblique angles

5.5.2 Multiplier-Less Smooth and Paeth Modes

Smooth Predictor: Generates gradients using quadratic interpolation. Hardware uses **Parallel Multiple Constant Multiplication (PMCM)** units—hardwired shift-add trees for the specific weights in the AV1 spec. This "multiplier-less" strategy significantly reduces dynamic power and gate count.

Paeth Predictor: Mimics local gradients. The logic calculates:

```
base = Top + Left - TopLeft
pLeft = |base - Left|
pTop = |base - Top|
pTopLeft = |base - TopLeft|
Prediction = argmin(pLeft, pTop, pTopLeft)
```

This entire decision tree is implemented in combinational logic, generating a whole block of Paeth prediction in a single clock cycle.

5.6 In-Loop Filtering: The Post-Processing Pipeline

The most computationally intensive part of AV1—and the area requiring the most specialized hardware—is the In-Loop Filtering stage. These filters remove artifacts and restore detail, but require reading and writing the entire frame.

5.6.1 Deblocking Filter

The Deblocking Filter Unit smooths edges between blocks. It operates on superblock boundaries using a small internal SRAM buffer for boundary pixels. Standard in all video codecs, but AV1's larger superblocks require wider filter support.

5.6.2 CDEF: Constrained Directional Enhancement Filter

CDEF is a novel AV1 tool designed to remove "ringing" artifacts near sharp edges while preserving the edges themselves. It presents unique hardware challenges:

Direction Search Unit:

- For every 8×8 block, hardware determines dominant edge direction
- Calculates variance of pixels along **8 candidate directions**
- Uses simplified integer math (not floating-point variance) to save power

Non-Linear Filtering:

- Once direction is identified, a non-linear low-pass filter is applied
- Filter taps pixels aligned with the edge direction
- Preserves edge sharpness while smoothing perpendicular noise

Line Buffering:

- CDEF requires access to a 12-tap window (current block + neighbors)
- Hardware maintains a dedicated CDEF Line Buffer
- For 4K video: buffer must store several full rows of the frame
- **Significant on-chip area trade-off in ASIC design**

Silicon Cost:

- Dedicated CDEF hardware block: $\sim$**185,000 logic gates**
- Substantial portion of decoder's logic budget
- **Vision Utility:** This is essentially a hardware Canny edge detector running "for free" alongside the codec

5.6.3 Loop Restoration: Machine-Learning-Derived Filters

Loop Restoration is the final pipeline stage, using ML-derived filters to restore quality. It supports two modes:

Wiener Filter Hardware:

- Implements separable convolution with coefficients signaled in the bitstream
- Reuses interpolation filter logic structure (saving area)
- Gate count: $\sim$**37,000 gates**

Self-Guided Filter (SGF) Hardware:

- More complex: preserves edges by calculating local mean and variance
- Requires **Integral Image logic** to calculate sums over rectangular regions in constant time

- Gate count: ∼**177,000 gates**
- Power dissipation: **60–115 mW** depending on configuration

5.6.4 Super-Resolution: Horizontal-Only Upscaling

AV1 allows frames to be coded at lower resolution and upscaled before display.

Hardware-Friendly Design Decision: The AV1 specification restricts normative super-resolution to **horizontal dimension only**. Vertical scaling would require buffering multiple full lines (expensive SRAM). Horizontal scaling is performed ”on the fly” within the processing pipeline of a single line, requiring minimal additional buffering.

5.7 Memory Hierarchy and Data Flow Optimization

In deep sub-micron processes (5nm, 7nm), **moving data costs more than computing on it**. DRAM access consumes orders of magnitude more energy than SRAM access. The architecture of an AV1 accelerator is fundamentally defined by its memory hierarchy.

5.7.1 Multi-Level Cache Hierarchy

To prevent the decoder from becoming memory-bound, hardware architects implement:

Cache Level	Size	Contents	Latency
L1 Reference Cache	16–64 KB	Current superblock’s prediction window	1 cycle
L2 Tile Cache	256 KB–1 MB	Frequently accessed reference regions	3–5 cycles
System Level Cache	6–18 MB	Reference frames (compressed)	10–20 cycles
DRAM	GB	Full frame buffer	100+ cycles

5.7.2 Frame Buffer Compression (FBC)

To reduce off-chip memory bandwidth, reference frames are stored in DRAM in a **compressed format**. The Memory Management Unit contains FBC decoder logic that decompresses blocks on-the-fly as they’re fetched.

- **Bandwidth Reduction:** 30–50%
- **Direct Power Savings:** Fewer DRAM transactions = lower memory controller power

5.7.3 The Scout’s Memory Advantage

The Scout sensor operates at a fundamentally different scale:

Parameter	Full 4K Encode	Scout (VGA)
Frame size	12.4 MB	460 KB
Reference frames	7× = 87 MB	1× = 460 KB
Total working set	∼100 MB	∼1 MB
Fits in SLC?	×(DRAM required)	✓(On-chip)

Zero DRAM transactions = minimal power. This is why the 80 mW VPU target is conservative. An optimized ME-only loop on 4nm silicon could operate at **10–30 mW**.

5.7.4 Tile-Based Parallelism

AV1 supports splitting frames into independent Tiles:

- **Parallel Cores:** High-end decoders (discrete GPUs) use multi-core architectures, each core processing a Tile

- **No Synchronization:** Tiles have no dependencies (except loop filtering boundaries)
- **Interconnect:** High-speed Network-on-Chip manages data flow between Tile cores and shared Frame Buffer

5.8 Power Management: The ”Low Power” Secret

5.8.1 Fine-Grained Clock Gating

AV1's many optional tools enable aggressive power management:

- If a superblock doesn't use Warped Motion $\rightarrow$ **clock to Warped Motion Unit is gated**
- If CDEF is disabled for a frame $\rightarrow$ **CDEF logic draws zero dynamic power**
- The controller monitors mode decisions and gates inactive units **cycle-by-cycle**

For ”Scout” mode (ME-only), the hardware can power down:

- Entropy coder (not generating bitstream)
- Reconstruction path (not reconstructing pixels)
- Loop filters (not deblocking/restoring)
- Transform engine (not processing residuals)

Result: Only the ME engine, MV cache, and control logic remain active.

5.8.2 Voltage Islands

In modern SoCs, the Media Engine resides in its own power domain:

- **Decoupled Voltage:** Even if GPU 3D cores run at high voltage for gaming, Media Engine operates at lower, optimal voltage for video
- **Dark Silicon:** During video playback, the massive GPU/CPU die can be powered down, leaving only the small, efficient Media Engine active

This is why a smartphone can play AV1 video for hours but would drain its battery in minutes doing software decode.

5.8.3 The Dark Silicon Opportunity

Modern SoCs face a fundamental constraint: **we can fabricate more transistors than we can power simultaneously**. A flagship mobile SoC contains $\sim$20 billion transistors, but thermal limits allow only a fraction to operate at full speed concurrently. This creates vast regions of ”dark silicon”—transistors that must remain idle to prevent thermal runaway.

The Conventional Problem:

Subsystem	Typical TDP	Duty Cycle	”Dark” When
CPU cluster	5-8 W	Variable	GPU-heavy games
GPU	6-10 W	Variable	CPU-heavy tasks
NPU	3-5 W	Variable	Non-ML workloads
VPU	0.1-0.3 W	Variable	No video

The Scout Insight:

The AV1 Retina exploits dark silicon by **keeping the massive compute blocks dark while running the tiny VPU**:

- GPU (billions of transistors) $\rightarrow$ **OFF** (100% dark)
- NPU (billions of transistors) $\rightarrow$ **OFF** (100% dark)

- CPU (mostly) $\rightarrow$ **OFF** (deep sleep)
- VPU ME engine ($\sim$500K transistors) $\rightarrow$ **ON** (fully lit)

Power Density:

$$\text{Scout Power Density} = \frac{140 \text{ mW}}{\sim 0.5 \text{ mm}^2} \approx 0.28 \text{ W/mm}^2$$

$$\text{GPU Power Density} = \frac{8000 \text{ mW}}{\sim 20 \text{ mm}^2} \approx 0.4 \text{ W/mm}^2$$

But the Scout area is tiny, so total thermal load is negligible. **The Scout can run indefinitely without thermal throttling**, while continuous GPU inference would thermal-throttle within minutes.

The Message to Silicon Vendors: You've already paid the area cost for the VPU. Expose it as a "Vision Mode" API and unlock always-on perception without touching your thermal budget.

6 Platform Analysis: The Silicon Landscape

The viability of the AV1 Retina depends on the hardware capabilities of real-world silicon. We evaluated three architectures representing the edge, the gap, and the cloud—analyzing not just specifications but the architectural decisions that enable or prevent Physics-First vision.

6.1 Google Tensor G5: The Ideal Scout

The Tensor G5 (Pixel 10 series) represents Google's first fully custom SoC, manufactured on TSMC's 3nm (N3E) process. This is a definitive break from the Samsung Exynos lineage that defined previous Tensor generations. Crucially for our analysis, Google has discarded its previous semi-custom "BigWave" video block in favor of a licensed, high-performance IP from Chips&Media: the **WAVE677DV**.

6.1.1 The WAVE677DV Core: Architecture

The integration of the WAVE677DV is the defining feature that positions the Tensor G5 as the premier mobile platform for the AV1 Retina.

Specification	WAVE677DV Capability
Codec Support	AV1, HEVC, H.264, VP9 (encode + decode)
Max Resolution	8K60 or 4K120
Superblock Size	Full 128×128 support
Bit Depth	8-bit, 10-bit, 12-bit HDR
Architecture	Dual-core video codec IP
Process	Optimized for 5nm/3nm nodes

6.1.2 Hardware Motion Estimation

The WAVE677DV implements a sophisticated hardware Motion Estimation engine capable of handling AV1's extensive reference frame pool:

- **Hierarchical Search:** Coarse search on downscaled frames followed by refinement on full resolution
- **7-Reference Support:** Hardware can simultaneously search across all AV1 reference frames
- **Vector Accessibility:** Through Android's MediaCodec API with Google's vendor extensions, developers can access encoding statistics including MVs

Scout Utility: The Tensor G5 can function as a "blind" motion sensor—analyzing the optical flow of the world in the VPU while the GPU and NPU remain in deep sleep.

6.1.3 CFrame: Solving the Memory Bandwidth Crisis

One of the hidden costs of video processing is DRAM access. Fetching reference frames for motion estimation consumes significant power. The WAVE677DV integrates **CFrame**, Chips&Media's proprietary lossless frame buffer compression.

Metric	Without CFrame	With CFrame
Reference frame bandwidth	100%	∼50%
LPDDR5X wake frequency	High	Reduced by half
Power for reference fetch	∼200 mW	∼100 mW

Impact for Always-On Scout: CFrame means the ME engine can compare frames while waking the memory bus half as often, directly contributing to the milliwatt-scale operation target.

6.1.4 Custom ISP Integration

Google has replaced the Samsung ISP with a fully custom Google ISP, enabling tight coupling with the VPU:

- **Direct Hardware Link:** The ISP and VPU likely share a direct hardware path or shared SRAM buffer
- **Minimal Handover Latency:** The "reflex arc" from motion detection (VPU) to high-res capture (ISP) can be minimized
- **Software Control:** Google controls the entire stack (Android + HAL + silicon), enabling optimizations impossible on third-party platforms

6.1.5 Tensor G5 Verdict

Role: Ideal Edge Scout

The Tensor G5 is architecturally ready to implement the full AV1 Retina. It can generate high-fidelity physics data (Motion Vectors, Partition Trees, CDEF state) entirely in hardware while keeping the GPU/NPU asleep. The combination of native AV1 encoding, CFrame compression, and custom ISP integration makes it the reference platform for edge-based Scout sensors.

6.2 Qualcomm Snapdragon 8 Elite: The Strategic Gap

The Snapdragon 8 Elite (SM8750) is a powerhouse of raw computation—second-generation Oryon CPU cores (4.32 GHz), sliced Adreno 830 GPU, and Hexagon NPU. However, a forensic examination of its multimedia specifications reveals a critical strategic divergence that handicaps its utility as an AV1 Retina.

6.2.1 The Missing Encoder

While the laptop-class Snapdragon X Elite explicitly boasts "AV1 encode and decode," the mobile Snapdragon 8 Elite conspicuously **omits AV1 encoding**:

Capability	Snapdragon 8 Elite	Snapdragon X Elite (Laptop)
AV1 Decode	✓Hardware (8K60)	✓Hardware
AV1 Encode	**✗ NOT SUPPORTED**	✓Hardware
Primary Encode	APV + HEVC	AV1 + HEVC

The product briefs list "Hardware-accelerated H.265, VP9, AV1 decoder" but only "HEVC" and "Dolby Vision" for capture/encoding. Analysis of leaked datasheets confirms: the Adreno VPU supports 8K60 AV1 decoding (consumption), but does not include a hardware AV1 encoder block.

6.2.2 The Pivot to APV

Qualcomm has placed its bet on the **Advanced Professional Video (APV)** codec for high-fidelity capture:

Property	APV	AV1
Frame Type	**Intra-only**	Inter + Intra
Motion Vectors	✗ None	✓Full optical flow
Compression Ratio	Lower (ProRes-like)	Higher
Use Case	Professional video capture	Streaming, storage
Royalties	Free	Free

The Fatal Flaw: Because APV is intra-frame-only, it **does not generate Motion Vectors**. Each frame is encoded independently without referencing past or future frames. A Scout system on Snapdragon 8 Elite using APV would be completely blind to motion.

6.2.3 The HEVC Fallback

The chip retains a powerful HEVC (H.265) encoder, which does generate motion vectors. However:

Limitation	HEVC	AV1
Max CTU Size	64×64	128×128
Partition Types	Square only	Square + T-shaped
Reference Frames	4	7
CDEF	✗	✓

A Physics-First system on Snapdragon 8 Elite must use HEVC, offering a **lower-resolution "compound eye"** compared to Tensor G5's native AV1.

6.2.4 Alternative Paths

Qualcomm does expose some motion data through proprietary APIs:

- **Hexagon DSP SDK:** "Video Analytic Stats" for Electronic Image Stabilization (EIS)
- **QMediaExtensions:** `KEY_ROI_INFO_TYPE` and `KEY_Enc_Stat` for encoding statistics

These provide proprietary paths to motion data, but they're not the clean, standardized AV1 pipeline.

6.2.5 Snapdragon 8 Elite Verdict

Role: Compromised Scout

The Snapdragon 8 Elite can perform Physics-First vision, but only through legacy HEVC or proprietary APIs. It misses AV1's superior partitioning granularity (128×128), advanced CDEF filtering, and 7-reference motion tracking. For edge Scout deployment, the Tensor G5 is the superior choice; for Snapdragon, accept the HEVC proxy or wait for the next generation.

6.3 NVIDIA Blackwell: The Ultimate Sniper

If edge devices are the Scouts, NVIDIA's Blackwell (RTX 50-series) and Ada Lovelace (RTX 40-series) GPUs are the Snipers—and the central brain capable of processing data from thousands of endpoints.

6.3.1 NVENC Evolution: Gen 8 to Gen 9

6.3.2 Split Frame Encoding (SFE)

SFE is a critical innovation for high-resolution vision systems:

Mechanism:

Generation	Architecture	Key Features
Gen 7	Turing/Ampere	H.264/HEVC encode; Optical Flow Accelerator (NVOFA)
Gen 8	Ada Lovelace	**AV1 hardware encode**; Split Frame Encoding; Dual encoders
Gen 9	Blackwell	AV1 UHQ mode; 4:2:2 10-bit; Lookahead for AV1; MV-HEVC stereo

1. Driver splits 4K/8K input frame into horizontal strips
2. Each strip fed to separate NVENC engine (high-end cards have dual encoders)
3. Engines encode strips in parallel
4. Bitstream stitched together

Latency Impact:

- 8K frame encoding latency reduced by $\sim$50%
- Enables faster reaction to incoming high-fidelity data triggered by remote Scout

6.3.3 The Motion Estimation Only API

NVIDIA provides a specific API feature that validates the AV1 Retina concept: `NvEncRunMotionEstimationOnly`.

```
// Setup ME-only mode
NV_ENC_INITIALIZE_PARAMS initParams = {0};
initParams.encodeConfig->frameIntervalP = 1;
initParams.encodeConfig->gopLength = NVENC_INFINITE_GOPLENGTH;

// Allocate MV output buffer
NV_ENC_CREATE_MV_BUFFER mvBufferParams = {0};
NvEncCreateMVBuffer(encoder, &mvBufferParams);

// Run ME-only pass (no bitstream generation)
NV_ENC_MEONLY_PARAMS meParams = {0};
meParams.inputBuffer = inputFrame;
meParams.referenceFrame = refFrame;
meParams.mvBuffer = mvBufferParams.mvBuffer;
NvEncRunMotionEstimationOnly(encoder, &meParams);

// Output structure per macroblock
typedef struct {
    int16_t mv_x;      // Motion vector X (1/4 pixel precision)
    int16_t mv_y;      // Motion vector Y (1/4 pixel precision)
    uint32_t sad;      // Sum of Absolute Differences
    uint8_t mbType;    // Macroblock type/partition
} NV_ENC_MV_DATA;
```

Function: This API bypasses the entire reconstruction, entropy coding, and bitstream generation stages. The hardware runs only the Motion Estimation search.

Output: Buffer containing Motion Vectors (direction + magnitude) and SAD cost for each macroblock.

SDK Support:

- SDK 12.1+: H.264 and HEVC ME-only
- SDK 13.0+: Extended for AV1 hardware block; MV-HEVC stereoscopic optimizations

6.3.4 Optical Flow Accelerator (NVOFA)

Distinct from the NVENC ME engine, NVIDIA GPUs include a dedicated Optical Flow Accelerator:

Property	NVENC ME	NVOFA
Purpose	Compression-optimal matching	Vision-accurate flow
Granularity	16×16 or 8×8	Down to 4×4
Robustness	Optimized for bitrate	Robust to intensity changes
Independence	Part of encoder	Separate fixed-function block

NVOFA is a premium "Scout" option, providing higher-fidelity optical flow for computer vision tasks. It operates independently of the encoder and CUDA cores.

6.3.5 Massive Parallelism for Distributed Scouts

A single Blackwell GPU can ingest Motion Vector streams from hundreds of edge devices:

- **Ingestion:** MV data is $\sim$1KB per frame (vs. $\sim$10MB for raw video)
- **Aggregation:** GPU monitors "Which of my 500 cameras sees unusual motion?"
- **Semantic Trigger:** Only when Scout anomaly detected does GPU run full inference
- **Throughput:** Thousands of MV streams analyzed at negligible compute cost

6.3.6 Blackwell Verdict

Role: Centralized Sniper / Cloud Backend

Blackwell is the ideal backend for a distributed Scout network. With Split Frame Encoding, exposed ME-Only APIs, and dedicated Optical Flow Accelerator, it can:

1. Aggregate physics data from edge devices
2. Apply semantic intelligence only where needed
3. Coordinate responses across the sensor network

6.4 Comparative Summary

Capability	Tensor G5	Snapdragon 8 Elite	NVIDIA Blackwell
Role	Edge Scout	Compromised	Cloud Sniper
AV1 Hardware Encode	✓(WAVE677DV)	✗	✓(NVENC Gen 9)
Motion Vector Access	VPU (Hardware)	HEVC only	ME-Only API
Max Partition Size	128×128	64×64 (HEVC)	128×128
Reference Frames	7	4 (HEVC)	7
CDEF / Edge Filter	✓Hardware	✗	✓Hardware
Bandwidth Optimization	CFrame ($\sim$50%)	UBWC ($\sim$50%)	FBC (30-50%)
Power Profile	$\sim$140 mW (Watchdog)	High (CPU fallback)	$\sim$300W (System)
Optical Flow Accelerator	✗	✗	✓(NVOFA)
Ideal Use Case	Always-on edge sensor	Fallback/legacy	Centralized hub

6.5 The Ecosystem Gap

The platform analysis reveals a critical industry pattern:

- **AV1 Decode:** Ubiquitous (all platforms)
- **AV1 Encode:** Rare on mobile (only Tensor G-series, some MediaTek)

- **ME-Only API:** Available on desktop (NVIDIA), absent on mobile

This creates a strategic opportunity: **the first mobile platform to expose a clean "Vision Mode" API for its AV1 encoder gains a significant advantage in edge AI applications.**

Google, with its control over Android + Tensor + Pixel Camera, is best positioned to close this gap. Full realization of the AV1 Retina will require industry cooperation—standardized APIs that expose Motion Vectors, partition trees, and CDEF state as first-class outputs. The hardware computes this data already; the missing piece is software access.

7 Validation: Power, Latency, and Bandwidth

The AV1 Retina makes specific quantitative claims. This section validates each against hardware data, academic research, and industry benchmarks.

7.1 Power Validation

Claim: The Scout can operate in a "Watchdog" loop at ∼140 mW (50 mW sensor + 80 mW VPU + 10 mW MCU).

7.1.1 Motion Estimation Power: Academic Evidence

Research on hardware Motion Estimation architectures provides direct power measurements:

Study / Architecture	Resolution	Power	Process Node	Notes
65nm H.264 ME core	CIF (352×288)	2.13 mW	65nm	Academic ASIC
4K UHD ME architecture	3840×2160 @ 60fps	107.76 mW	28nm	Industry reference
Hierarchical ME (TI)	1080p	45 mW	40nm	Commercial IP
Scaled VGA estimate	640×480	∼4 mW	28nm	Linear scaling from 4K

Scaling Analysis:

4K resolution has $(3840 \times 2160) = 8.3M$ pixels per frame. VGA resolution has $(640 \times 480) = 307K$ pixels per frame.

$$\text{Pixel Ratio} = \frac{8.3M}{307K} \approx 27\times$$

If ME power scales linearly with pixels (reasonable for throughput-limited designs):

$$P_{VGA} = \frac{107.76 \text{ mW}}{27} \approx 4 \text{ mW}$$

Modern process nodes (4nm vs. 28nm) provide additional savings:

- Dynamic power scales with $C \cdot V^2 \cdot f$
- Voltage reduction: ∼0.9V → ∼0.65V (roughly 2× reduction in V^2)
- Capacitance reduction: ∼3× from process scaling

Conservative Estimate: Even accounting for control logic, clock distribution, and SRAM access overhead, an ME-only VPU at VGA resolution on 4nm silicon should consume **10–30 mW**. The 80 mW budget provides substantial margin.

Encoder Stage	Typical Power	In Scout Mode?
Motion Estimation	20–40 mW	✓Required
Intra Prediction	15–30 mW	✗ Skipped
Transform (DCT)	20–40 mW	✗ Skipped
Quantization	10–20 mW	✗ Skipped
Entropy Coding (CABAC)	30–60 mW	✗ Skipped
Reconstruction	20–40 mW	✗ Skipped
Deblocking Filter	15–30 mW	✗ Skipped
CDEF	60–115 mW	✗ Skipped*
Loop Restoration	40–80 mW	✗ Skipped
Full Encode	**230–455 mW**	
ME-Only	**20–40 mW**	

7.1.2 The Power Consumers We Eliminate

The key to Scout efficiency is what we **don't** run:

*CDEF can optionally be enabled if edge detection is needed.

7.1.3 Memory Power: The SRAM Advantage

DRAM vs. SRAM Power:

Memory Type	Access Energy	Typical Power @ 30fps 4K
LPDDR5X	$\sim$20 pJ/bit	400–600 mW
On-chip SRAM	$\sim$0.5 pJ/bit	10–30 mW

The Scout operates at VGA resolution:

- Frame size: $640 \times 480 \times 1.5 = 460$ KB (YUV420)
- Working set: Current frame + reference frame + MV buffer $\approx$ **1 MB**
- Modern SoC SLC: **6–18 MB**

Result: The entire Scout working set fits in on-chip SRAM. Zero DRAM transactions eliminate the largest power consumer in video processing.

7.1.4 Power Validation Conclusion

The 140 mW target is **highly conservative**. Real-world implementations could achieve:

- Optimistic: 30–50 mW (aggressive clock gating, process optimization)
- Conservative: 100–150 mW (margin for control, I/O, thermal)

7.2 Latency Validation

Claim: The Scout can achieve sub-5 ms "reflex arc" latency from sensor input to motor command.

7.2.1 Encoding Latency Benchmarks

Parsec, a low-latency cloud gaming provider, has extensively benchmarked hardware encoders:

7.2.2 Scaling to Scout Resolution

Encoding latency scales roughly with pixel count (throughput-limited):

Platform	Resolution	Encoding Latency (Median)
NVIDIA NVENC (Turing)	1080p	5.8 ms
NVIDIA NVENC (Ada)	1080p	4.5 ms
AMD VCN 3.0	1080p	6.2 ms
Intel QSV (Arc)	1080p	5.5 ms

$$\text{Latency}_{VGA} = \text{Latency}_{1080p} \times \frac{\text{Pixels}_{VGA}}{\text{Pixels}_{1080p}}$$

$$= 5.8 \text{ ms} \times \frac{640 \times 480}{1920 \times 1080} = 5.8 \text{ ms} \times 0.148 \approx 0.86 \text{ ms}$$

ME-Only Additional Savings: Since we skip entropy coding, reconstruction, and loop filtering, actual ME-only latency is even lower. Estimated: $<$**0.5 ms**.

7.2.3 Pipeline Latency Breakdown

Stage	Latency	Notes
Sensor exposure	8–16 ms	60–120 fps capture; can use rolling readout
Sensor readout	2–5 ms	MIPI CSI-2 transfer
VPU ME processing	0.5–1 ms	Hardware ME at VGA
MCU threshold check	$<$0.1 ms	Simple comparison logic
Motor command	0.5–1 ms	PWM/CAN bus latency
Total Reflex Arc	**11–23 ms**	Sensor-limited
Post-capture to action	$<$**3 ms**	VPU + MCU + motor

7.2.4 Slice-Based Streaming

Hardware encoders support **Slice** or **Tile** based processing. The ME engine can output Motion Vectors for the top slice of the frame before the bottom slice is captured.

"Racing the Beam": With a 120 fps sensor (8.3 ms frame time) and slice-based ME, the system can detect motion in the top half of the frame in $\sim$1 ms, while the bottom half is still being read out.

7.2.5 Zero-Copy Memory Path

The primary latency threat is **software overhead**, not hardware.

Standard Path: Sensor $\rightarrow$ VPU Memory $\rightarrow$ **Copy to System RAM** $\rightarrow$ **OS Scheduler** $\rightarrow$ CPU Process $\rightarrow$ Motor

Zero-Copy Path: Sensor $\rightarrow$ VPU Memory $\rightarrow$ **Direct MCU Read** $\rightarrow$ Motor

Platforms supporting zero-copy:

- NVIDIA Jetson: `NvBufSurface` API
- Linux V4L2: `VIDIOC_EXPBUF` + DMA-BUF
- Raspberry Pi: MMAL buffer sharing

7.2.6 Latency Validation Conclusion

Sub-5 ms reflex latency **from capture completion to motor command** is achievable with:

- Hardware ME-only mode

- Zero-copy memory architecture
- Direct MCU $\rightarrow$ motor path (bypassing OS)

Total sensor-to-action latency is dominated by sensor frame time (8–16 ms at 60–120 fps), not processing.

7.3 Bandwidth Validation

Claim: The Scout operates entirely within on-chip memory bandwidth, requiring zero DRAM transactions.

7.3.1 Scout Bandwidth Requirements

Data Flow	Size per Frame	Rate @ 60fps
Input frame (VGA YUV420)	460 KB	27.6 MB/s
Reference frame	460 KB	27.6 MB/s
Motion Vector output	$\sim$10 KB	0.6 MB/s
Total		$\sim$**56 MB/s**

7.3.2 On-Chip SRAM Bandwidth

Cache Level	Typical Bandwidth	Sufficient?
L1 Cache	100–500 GB/s	✓$\sim$1000$\times$ margin
L2 Cache	50–200 GB/s	✓$\sim$500$\times$ margin
System Level Cache	20–80 GB/s	✓$\sim$200$\times$ margin

7.3.3 Sniper Bandwidth (When Active)

Data Flow	Size per Frame	Rate @ 30fps
Input frame (4K YUV420)	12.4 MB	372 MB/s
Reference frames (7$\times$)	87 MB	2.6 GB/s
Total		$\sim$**3 GB/s**

This exceeds on-chip capacity, requiring DRAM—hence the high power. Frame Buffer Compression (CFrame, UBWC, FBC) reduces this by 30–50%.

7.3.4 Bandwidth Validation Conclusion

The Scout's VGA resolution creates a **qualitatively different memory regime**:

- Working set: $\sim$1 MB (fits in SLC)
- Bandwidth: $\sim$56 MB/s (trivial for on-chip SRAM)
- DRAM transactions: Zero

This is the fundamental efficiency lever: **staying on-chip eliminates the largest power consumer**.

7.4 Comparison: Neural Network Inference

For context, here's what the traditional approach costs:

The Scout (ME-only) achieves **motion detection**—not classification—at:

- Power: 80–140 mW (comparable to MobileNetV3)

Model	Resolution	MACs/frame	Power @ 30fps	Latency
YOLOv8-nano	640×640	1.9G	500–800 mW	15–30 ms
YOLOv8-small	640×640	7.2G	1–2 W	25–50 ms
YOLOv8-medium	640×640	20G	2–4 W	40–80 ms
MobileNetV3 (class.)	224×224	0.2G	100–300 mW	5–15 ms

- Latency: $<$1 ms (5–15$\times$ faster than inference)
- Output: Physics data (motion vectors), not semantic labels

The power and latency are similar to the smallest neural networks, but the Scout provides **continuous monitoring** without the duty-cycle limitations of inference.

8 Strategic Implications

8.1 Privacy by Architecture, Not by Policy

Motion Vectors and Partition maps are **mathematically non-invertible representations**. Unlike encrypted or blurred video (which can potentially be decrypted or enhanced), MV fields contain insufficient information to reconstruct the original scene.

8.1.1 The Information-Theoretic Argument

Representation	Data per Frame (VGA)	Bits per Pixel	Invertible?
Raw RGB	921 KB	24	✓Trivially
YUV420	460 KB	12	✓Trivially
JPEG (Q=90)	∼50 KB	∼1.3	✓Perceptually
Motion Vectors	∼10 KB	∼0.26	✗ **Impossible**
Partition Map	∼2 KB	∼0.05	✗ **Impossible**

Why Inversion is Impossible:

Motion Vectors describe *displacement*, not *content*. Consider two frames:

- Frame A: A white circle on black background
- Frame B: The circle moved 10 pixels right

The Motion Vector field contains: ”Region X moved (10, 0)”

Given *only* the MV field, you cannot determine:

- The circle was white (could be any color)
- The background was black (could be any color)
- The object was circular (could be any shape with the same motion)
- What the scene depicts (could be anything moving that way)

Even with sophisticated ML attacks, the best an adversary could infer is ”something roughly this size moved this direction.” That’s the point—it’s sufficient for physics detection but insufficient for identification.

8.1.2 Regulatory and Deployment Advantages

Practical Implication: A smart home device operating in Scout-only mode:

- Detects intruders, falls, unusual activity patterns
- **Never** captures faces, reads documents, or records identifiable behavior

Regulatory Domain	Scout Data	Sniper Data
GDPR (EU)	Potentially exempt (no PII)	Full compliance required
CCPA (California)	Minimal concern	Consent required
BIPA (Illinois)	N/A (no biometrics)	Strict requirements
HIPAA (Healthcare)	Non-PHI	PHI if identifiable

- Can be deployed in privacy-sensitive contexts (bedrooms, bathrooms, children's spaces) where cameras are socially unacceptable

The Sniper is Opt-In: The high-resolution, identifiable capture activates *only* when physics demand it—and that activation can require explicit user consent, multi-factor authentication, or policy approval.

This is **privacy by architecture**—not a promise that can be broken by a software update, but a mathematical impossibility enforced by information theory.

8.2 Latency Reduction for Safety-Critical Systems

In robotics and autonomous vehicles, reaction time is survival time.

System	Latency	Use Case
Human visual reaction	$\sim$250 ms	Baseline
GPU inference loop	30–100 ms	Object detection
AV1 Retina reflex	$<$**5 ms**	Collision avoidance

A drone can initiate evasive maneuvers in the time it takes a GPU to load a neural network. The "reflex arc" doesn't need to identify the threat—it needs to move.

8.3 Scaling: The Compound Eye Network

The Scout & Sniper architecture scales naturally:

- **1 Sniper : N Scouts** — A single NVIDIA GPU can monitor MV streams from hundreds of edge sensors
- **Hierarchical Filtering** — Only anomalous physics data propagates to the cloud
- **Bandwidth Reduction** — Transmitting Motion Vectors (KB/s) instead of video (MB/s)

This enables massive sensor networks (smart city, industrial IoT) at a fraction of the bandwidth and compute cost.

9 The Software and API Landscape

9.1 The Access Problem

The hardware is ready. The software is the bottleneck.

Standard APIs (Android MediaCodec, Windows Media Foundation) are designed for **media playback**, not machine vision. They hide the "messy" internal physics data to present a clean bitstream.

API	Motion Vector Access	Partition Access	Notes
Android MediaCodec	✗ (standard)	✗	Vendor extensions required
NVIDIA Video Codec SDK	✓(H.264/HEVC)	✓	`NvEncRunMotionEstimationOnly`
Intel Media SDK	Partial	✗	Statistics available for AV1
Linux V4L2 (Stateless)	✓	✓	Low-level access on RK3588, etc.

9.2 Vendor-Specific Paths

NVIDIA (Best Documented):

```
// Allocate MV buffer
NvEncCreateMVBuffer(encoder, &mvBuffer);

// Run ME-only pass
NvEncRunMotionEstimationOnly(encoder, &meParams);

// Read output: MV (x, y) + SAD for each macroblock
NV_ENC_H264_MV_DATA* mvData = mvBuffer.outputPtr;
```

Qualcomm (Proprietary):

- `QMediaExtensions` class exposes `KEY_ROI_INFO_TYPE` and `KEY_Enc_Stat`
- Motion data available through Hexagon DSP for EIS
- Limited to HEVC on Snapdragon 8 Elite (no AV1 encode)

Google Tensor:

- WAVE677DV likely exposes stats through HAL extensions
- Emerging documentation; expect deeper access for Pixel Camera features

9.3 The H.264/HEVC Proxy Pivot

Given the scarcity of AV1 hardware encoders on current mobile platforms, practical deployment requires a **codec-agnostic** approach:

1. Use H.264 or HEVC hardware encoder as the Scout
2. Motion Vectors from H.264 are physically equivalent to AV1 (same optical flow)
3. Accept reduced partition granularity (64×64 vs. 128×128)
4. Migrate to AV1 as hardware matures (Snapdragon 8 Gen 4, etc.)

The physics don't change based on the codec. An object moving left generates a "left" Motion Vector whether encoded as H.264, HEVC, or AV1.

10 Conclusion

The AV1 Retina represents a fundamental shift from **Pixel-First** to **Physics-First** computer vision.

The Core Insight: Modern video codecs are not merely compression tools—they are hard-wired, low-power physics engines. The Motion Estimation, Block Partitioning, and Directional Filtering operations that codecs perform to achieve compression are precisely the operations that machine vision systems pay billions of FLOPs to rediscover.

The Architecture: By repurposing the hardware VPU as a "Scout" sensor, we achieve:

- **25–40$\times$ power reduction** over always-inference loops
- **Sub-5ms reflex latency** for safety-critical reactions
- **Privacy-preserving perception** through non-invertible representations

The Platforms:

- **Google Tensor G5** emerges as the ideal edge Scout—full AV1 hardware encoding, CFrame compression, custom ISP integration
- **Qualcomm Snapdragon 8 Elite** represents a strategic gap, omitting AV1 encode in favor of intra-frame APV

- **NVIDIA Blackwell** defines the Sniper role—massive parallelism, exposed ME-Only APIs, centralized physics aggregation

The Path Forward:

1. **Immediate:** Deploy H.264/HEVC-based Scouts on legacy silicon, and native AV1 Scouts on the Google Pixel 10 (Tensor G5)
2. **Near-term:** Expand to emerging AV1-encode-capable SoCs from other vendors
3. **Long-term:** Standardized "Vision Mode" APIs that expose Motion Vectors and Partitions as first-class outputs

Full implementation requires industry cooperation—silicon vendors exposing the physics data their hardware already computes, and platform developers defining standard representations for this output. The hardware exists today; the software abstraction layer does not.

The AV1 Retina is no longer a theory—it is a silicon reality waiting to be exploited.

A Glossary

Term	Definition
Motion Vector (MV)	A 2D vector describing pixel displacement between frames
SAD	Sum of Absolute Differences; a cost metric for block matching
Superblock	AV1's maximum coding unit (128×128 pixels)
CDEF	Constrained Directional Enhancement Filter; AV1's deringing filter
VPU	Video Processing Unit; fixed-function encode/decode hardware
ME-Only Mode	Encoder mode that outputs Motion Vectors without completing bitstream
Scout	Low-power sensor for physics detection (motion, structure)
Sniper	High-power sensor for semantic identification (classification)

B Power Budget Breakdown

B.1 Detailed Scout Power Model

Component	Subcomponent	Power (mW)	Notes
Sensor	Pixel array	30	VGA monochrome
	Readout	15	Global shutter
	Interface	5	MIPI CSI-2
VPU	ME Engine	40	SAD array + search logic
	MV Cache	20	SRAM access
	Control	10	State machine
	Clock Tree	10	Distribution overhead
MCU	Threshold logic	5	Simple comparisons
	Interrupt	5	Wake signal generation
Total		**140 mW**	Conservative estimate

B.2 Sniper Activation Cost

Event	Energy Cost	Duration	Notes
Wake Sniper sensor	50 mJ	100 ms	Sensor power-on + stabilization
ISP processing	100 mJ	33 ms	Single 4K frame
NPU inference	150 mJ	50 ms	Object detection model
Total per event	**300 mJ**	$\sim$200 ms	Amortized over hours of Scout watch