CHALMERS
UNIVERSITY OF TECHNOLOGY

An FPGA-Based Visual Processing System for Autonomous Driving under Extreme Lighting

Master's thesis in Embedded Electronic System Design

Chen Wu, Xingchen Wu

Department of Microtechnology and Nanoscience
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026

Master's thesis 2026

An FPGA-Based Visual Processing System for Autonomous Driving under Extreme Lighting

Chen Wu, Xingchen Wu

Department of Microtechnology and Nanoscience
Chalmers University of Technology
Gothenburg, Sweden 2026

An FPGA-Based Visual Processing System for Autonomous Driving under Extreme Lighting
Chen Wu, Xingchen Wu

Supervisor: Per Larsson-Edefors, MC2, Chalmers University of Technology
Examiner: Lena Peterson, MC2, Chalmers University of Technology

Master's Thesis 2026
Department of Microtechnology and Nanoscience
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Daytime and nighttime road scenes captured for visual comparison under challenging lighting conditions.

Typeset in LaTeX
Gothenburg, Sweden 2026

An FPGA-Based Visual Processing
System for Autonomous Driving
under Extreme Lighting
Chen Wu, Xingchen Wu
Department of Microtechnology and Nanoscience
Chalmers University of Technology

Abstract

Extreme lighting conditions degrade camera image quality and may cause fatal autonomous driving accidents through perception failures. This thesis presents an FPGA-based front-end visual processing system that mitigates such degradation in real time while preserving downstream model recognition accuracy. The system architecture contributes two innovations: the traditional image signal processor pipeline is pruned from over ten hardware modules to five essential modules based on literature consensus, and black-level correction, auto exposure gain, and localized bloom suppression are consolidated into a single Bayer-domain affine correction applied before demosaicing, severing the saturation-energy diffusion cascade at its source. Auto exposure control operates as an independent side channel with zero pixel-throughput overhead. Validation follows a task-driven paradigm using contrastive-learning-based style transfer, where downstream model accuracy rather than pixel-level metrics serves as the evaluation criterion. FPGA implementation achieves low resource consumption, deterministic fixed latency, and throughput far exceeding real-time requirements. Style-transfer-trained models match the accuracy of those trained on original data in both object detection and semantic segmentation tasks, and visual comparison against a professional action camera confirms comparable overall image quality under both daytime and nighttime conditions, while achieving substantially lower and deterministic pipeline latency.

Keywords: FPGA, image signal processor, extreme lighting, bloom suppression, autonomous driving, machine vision

Acknowledgements

We would like to express our sincere gratitude to our supervisor, Per Larsson-Edefors, for his attentive guidance, insightful technical feedback, and continuous support throughout this project. We are equally grateful to our examiner, Lena Peterson, for her valuable advice on the writing style, structure, and content of this thesis. Her detailed comments have significantly improved the clarity and quality of this report.
We also wish to thank our families and friends for their encouragement and patience during the course of this work.

Chen Wu, Xingchen Wu, Gothenburg, July 2026

Contents

List of Abbreviations

Abbreviation	Full Name
AE	Auto Exposure
AEC	Auto Exposure Control
AF	Auto Focus
AWB	Auto White Balance
BLC	Black Level Correction
BRAM	Block Random Access Memory
CCM	Color Correction Matrix
CFA	Color Filter Array
CLAHE	Contrast Limited Adaptive Histogram Equalization
CMOS	Complementary Metal-Oxide-Semiconductor
CSC	Color Space Conversion
CUT	Contrastive Unpaired Translation
DPC	Defective Pixel Correction
DSP	Digital Signal Processor (slice)
FPGA	Field-Programmable Gate Array
FPS	Frames Per Second
FSD	Full Self-Driving
FSM	Finite State Machine
HVS	Human Visual System
IIR	Infinite Impulse Response
IPU	Image Processing Unit
ISP	Image Signal Processor
LSC	Lens Shading Correction
LUT	Lookup Table
MIPI CSI-2	Mobile Industry Processor Interface Camera Serial Interface 2
PID	Proportional-Integral-Derivative
PSNR	Peak Signal-to-Noise Ratio
RAW	Raw Sensor Data
RGB	Red, Green, and Blue
RTL	Register-Transfer Level
SD	Secure Digital
SSIM	Structural Similarity Index Measure

1

Introduction

Camera-based vision is one of the most critical sensing modalities in autonomous driving. The quality of the input video stream is as important as the model architecture and training methodology in determining recognition accuracy and driving safety. Raw image data from camera sensors must be processed through an image signal processor (ISP) front-end pipeline and converted into a red, green, and blue (RGB) format acceptable to downstream perception models. The details of Bayer-format sensor data and demosaicing are introduced in Chapter 2; at this stage, the key point is that ISP output quality is a prerequisite for downstream performance.

Real-world scene conditions are far from ideal. Image degradation, namely the loss of visual information caused by physical or environmental disturbances during acquisition, manifests most severely in autonomous driving under extreme lighting. Intense direct sunlight drives sensor pixels to saturation, causing complete loss of texture in highlight regions. At night, oncoming headlights and street lamps produce large-area halos through internal lens reflections and sensor charge overflow. In backlit scenes, foreground objects are compressed into an extremely narrow code-value range where detail is submerged in quantization noise.

Such failures are not hypothetical. In 2016, a Tesla Model S in Autopilot mode collided with a white semi-trailer truck crossing a Florida highway, killing the driver, because strong sunlight prevented the vision system from distinguishing the white truck against the bright sky [1]. In 2023, a Tesla in full self-driving (FSD) mode struck and killed a pedestrian in Arizona during sunset backlighting [2]. Multiple owners have documented FSD misclassifying red traffic lights as yellow under direct sunlight glare [3]. These incidents reveal that extreme-lighting image degradation is not an aesthetic concern, but a perception failure that directly threatens lives.

These examples motivate a closer examination of how extreme lighting damages the image before it reaches the perception model. At a high level, bloom first hides scene content and then interacts with demosaicing and auto exposure control (AEC), which can further reduce usable detail in darker regions. The detailed degradation mechanism is introduced in Chapter 2, while the architectural implications are discussed in Chapter 6. This observation provides the core motivation for relocating gain control and bloom suppression to the Bayer domain, before demosaicing.

Beyond image quality, the system must also satisfy a stringent latency constraint. For camera-based indirect vision systems that replace physical rear-view mirrors, UNECE Regulation No. 46 [4] and the equivalent Chinese national standard GB 15084-2022 [5] specify a maximum end-to-end latency of 200 ms. Because this limit applies

to the entire camera-to-display chain, the ISP processing stage must consume only a small fraction of this budget to leave adequate margin for sensor exposure, encoding, transmission, and display. Deterministic, sub-millisecond ISP latency is therefore a hard design requirement rather than an optimization target.

While various ISP products exist for conventional human-viewing-oriented imaging, to the best of our knowledge, no ISP pipeline has been specifically optimized for downstream model inference. Existing work focuses on image enhancement algorithms themselves, on accelerating back-end vision tasks, or on platform-level system integration. However, few works simultaneously address front-end visual degradation under extreme lighting, deterministic low-latency streaming processing, field-programmable gate array (FPGA) hardware deployability, and downstream perception model compatibility within one unified framework. This thesis aims to fill this gap.

1.1 Related Work

Image enhancement has been extensively studied, with methods ranging from classical histogram equalization and Retinex-based decomposition [6], [7], [8] to deep learning approaches [9], [10], [11]. While deep methods achieve high visual quality, their computational cost hinders real-time embedded deployment [12]. The strategy for FPGA deployment replaces expensive operations with hardware-friendly approximations, and real-time implementations of several classical and learning-based methods have been demonstrated on FPGA [13], [14], [15], [16]. However, these works focus on improving image appearance for human viewing. They do not address the question of whether the processing benefits or harms downstream machine vision models.

This question has been investigated from the ISP pipeline perspective. A standard FPGA ISP pipeline can be fully pipelined at one pixel per clock cycle using line buffers and fixed-point arithmetic [17], [18]. However, the assumption that all traditional stages contribute value no longer holds when the output serves machine recognition. Dodge and Karam [19] first showed that human-oriented quality metrics correlate only weakly with neural network accuracy. Buckler et al. [20] then conducted a landmark ablation study demonstrating that among the many ISP stages, only demosaicing and gamma compression significantly affect computer vision performance. Subsequent work has quantified per-stage contributions, proposed dynamic parameter control, and applied reinforcement learning to automatically select optimal module subsets. For instance, saturation enhancement consistently receives a zero selection frequency [21], [22], [23]. A heterogeneous FPGA pipeline combining traditional processing with lightweight neural networks at quality-critical stages has also been demonstrated [24].

In parallel, the evaluation paradigm has shifted. Traditional ISP verification relies on pixel-level metrics such as peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) [25]. PSNR quantifies per-pixel reconstruction error; SSIM incorporates luminance, contrast, and structure comparisons to approximate perceived image similarity. Both were designed for human-viewing-oriented quality

assessment, and Dodge and Karam [19] showed that their correlation with downstream model accuracy is unreliable. The alternative, task-driven evaluation, validates ISP processing through downstream model accuracy rather than pixel-level fidelity [26]. At the hardware verification level, C++ reference model and register-transfer level (RTL) co-verification with automated bit-exact comparison were first demonstrated in the FPGA ISP domain by Hegarty et al. [27].

Taken together, these three lines of work, ISP pruning, task-driven evaluation, and hardware co-verification, provide the foundation for this thesis. However, no prior work simultaneously addresses front-end visual degradation under extreme lighting, deterministic fixed-latency streaming hardware, low FPGA resource consumption, and task-driven validation within a unified framework.

1.2 Goal and Contributions

This thesis presents a complete FPGA-based end-to-end solution spanning hardware architecture, validation methodology, and experimental evaluation. The scope is deliberately focused: the system targets front-end visual degradation caused by extreme lighting, specifically bloom, glare, and the associated exposure control instability. It does not address other adverse conditions such as rain, fog, or snow. Validation is performed on a single forward-facing camera; multi-camera surround-view coordination is outside the scope. The hardware implementation targets the Xilinx xc7z035 FPGA and assumes a Mobile Industry Processor Interface Camera Serial Interface 2 (MIPI CSI-2) sensor interface; portability to other FPGA families is not evaluated. The downstream effectiveness validation relies on style transfer as a proxy for real dual-camera capture, and the limitations of this approach are discussed in Chapter 6.

First, the traditional ISP is pruned from over ten modules to five essential modules based on the literature consensus established by Buckler et al. [20], Hansen et al. [21], and Wang et al. [23]. Modules serving only human visual appearance, including sharpening, denoising, saturation enhancement, contrast stretching, and contrast limited adaptive histogram equalization (CLAHE), are removed. The retained modules are further reduced through mathematical fusion into a streamlined three-stage pipeline with an AEC side channel.

Second, the core architectural innovation consolidates three traditionally separate operations into a single Bayer-domain module: black level correction (BLC) offset subtraction, AEC global gain multiplication, and bloom halo suppression. The critical design decision is to apply these operations before demosaicing. Linear gain scaling is compatible with interpolation, so Bayer-domain application preserves the effect of the corresponding RGB-domain gain while acting before saturated energy is diffused.

Third, the AEC gain computation is architecturally separated from the pixel data path as an independent side-channel control loop, contributing zero overhead to pixel throughput. The side channel collects a RAW-domain histogram, computes key percentiles during vertical blanking, and determines the next-frame gain through

a five-state finite state machine with priority-ordered decision logic. Dual-layer temporal filtering ensures inter-frame brightness stability.

Fourth, a style-transfer-based validation methodology uses contrastive unpaired translation (CUT) to learn imaging style differences between ISP pipelines from unpaired data, generating style-transferred images as a proxy for real captures. Since the translation model is a lossy mapping that cannot inject new valid visual information, comparable model accuracy between original and transferred training data supports the claim that ISP-induced style variations cause no substantive harm to downstream models.

Fifth, the complete system is implemented on a Xilinx xc7z035 FPGA device. Core ISP logic consumes under 3,000 LUTs and under 50 block random access memory (BRAM) tiles, with digital signal processor (DSP) usage in the low teens. It operates near 180 MHz and delivers per-frame processing times well under 5 ms for 1080p video, which is over an order of magnitude above real-time requirements. Experimental results demonstrate that detection and segmentation models trained on style-transferred data achieve accuracy comparable to those trained on original data. Visual comparison against a professional action camera shows comparable overall image quality under both daytime and nighttime conditions, while the system achieves substantially lower and deterministic pipeline latency.

1.3 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 covers the technical background: ISP architecture, literature evidence on module effectiveness for machine vision, the bloom degradation mechanism, key ISP modules, the hardware platform, and evaluation models and datasets. Chapter 3 describes the design methodology, the hardware verification workflow, and the downstream validation strategy. Chapter 4 details the system design: the two-step simplification, the three-stage pipeline architecture with integrated AEC side channel, the physical setup, and the downstream validation experiment configuration. Chapter 5 presents experimental results including FPGA synthesis, style transfer outputs, model accuracy comparisons, and visual comparison against a reference camera. Chapter 6 discusses Bayer-domain preprocessing, the validation strategy rationale, design philosophy trade-offs, streaming versus block-parallel processing, and resource efficiency. Chapter 7 concludes with a summary of results and future work directions.

2

Technical Background

Following the motivation introduced in Chapter 1, this chapter provides the technical foundation needed to explain how extreme lighting affects ISP output and downstream perception. Section 2.1 surveys the ISP pipeline and reviews the literature on module effectiveness for machine vision. Section 2.2 analyzes the bloom degradation mechanism. Section 2.3 examines the key ISP modules. Section 2.4 describes the hardware platform, including the FPGA device and image acquisition equipment. Section 2.5 introduces the validation tools, evaluation models, and benchmark datasets.

2.1 ISP Module Effectiveness for Machine Vision

An ISP is a specialized hardware or software pipeline that transforms raw sensor data from a camera into a viewable image. When light strikes a camera sensor, the raw output consists of millions of values representing light intensity alone. It is noisy, suffers from color imbalance, and is improperly exposed. The ISP corrects these issues in real time through a sequence of processing stages.

A complete traditional ISP pipeline, as illustrated in Figure 2.1, executes the following modules in sequence: BLC, lens shading correction (LSC), defective pixel correction (DPC), demosaicing, auto white balance (AWB), color correction matrix (CCM), noise reduction, gamma correction, and color space conversion (CSC). These modules are connected in series along the data flow direction, with feedback control provided by the 3A algorithms, namely auto exposure (AE), auto focus (AF), and AWB. Table 2.1 summarizes each stage and its primary function.

Traditional ISP pipelines are largely designed for human viewing. Many stages therefore reflect assumptions about human visual perception, such as nonlinear brightness response, perceived contrast, and lightness perception relative to surrounding regions [29], [7], [25]. These assumptions do not necessarily remain valid when the output is used by machine vision models. A visually pleasing image is not always the most informative input for downstream recognition.

Denoising is a clear example of this mismatch. It is useful for human viewing because random noise in flat regions such as sky and road surfaces is visually disturbing. For convolutional networks, however, aggressive denoising may remove fine texture and weak boundaries that are useful for recognition. AdaptiveISP reported a reinforcement-learning selection frequency of 0% for denoising across all tested

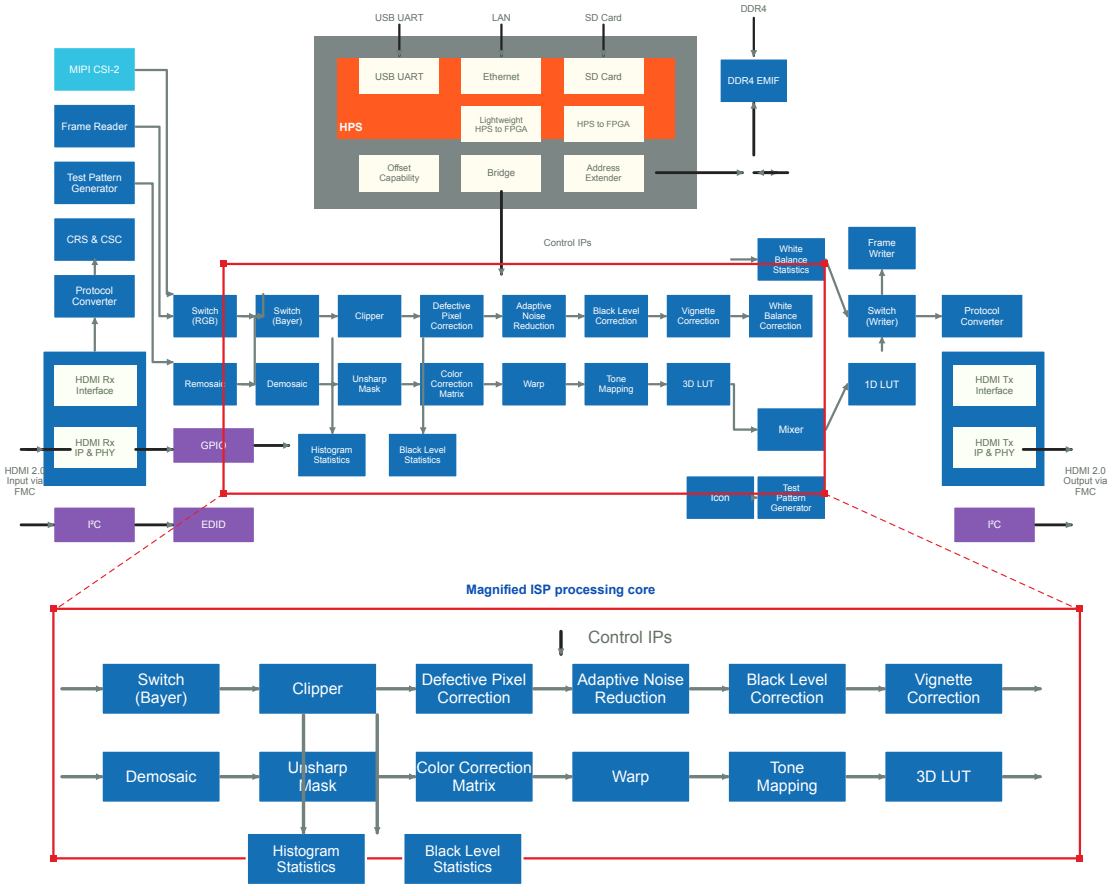

Figure 2.1: Traditional ISP pipeline. Reproduced from Intel white paper [28].

Table 2.1: Typical ISP pipeline stages.

Abbreviation	Primary Function
BLC	Subtracts sensor dark current offset
LSC	Compensates vignetting and color shading
DPC	Detects and replaces stuck/hot pixels
Demosaicing	Reconstructs full-color RGB from Bayer array
AWB	Normalizes per-channel gain for neutral gray
CCM	Maps sensor color space to sRGB
Noise reduction	Suppresses random and structured noise
Gamma correction	Applies nonlinear tone mapping
CSC	Converts RGB to YUV/YCbCr for encoding

conditions [23]. GenISP also showed that conventional denoising combined with brightness enhancement can create visually smooth but semantically misleading regions under low light, which may harm downstream detection [30].

Sharpening and saturation enhancement show similar limitations. Sharpening can improve perceived clarity, but it may also amplify noise and artificial edge halos. AdaptiveISP showed that sharpening can still be useful for accuracy-priority configurations, but its function is related to matching the spatial-frequency content expected by the downstream network rather than to human visual preference [23]. Saturation enhancement is even less suitable for machine vision, since artificially boosted colors may not correspond to real scene properties. Under low-light conditions, AdaptiveISP found that the model preferred desaturation rather than saturation enhancement [23].

CLAHE is a widely used local contrast enhancement method in image processing [31]. However, its tile-based histogram equalization and interpolation between local mappings are not well matched to a streaming FPGA ISP architecture. The method requires local histograms, tile-wise mapping tables, and multiple lookups per pixel, which increase storage and control complexity. In a resource-constrained and deterministic-latency FPGA pipeline, this cost is difficult to justify when the target is downstream recognition rather than human viewing.

Contrast stretching illustrates a more general limitation of global enhancement operations under bloom. When saturated regions occupy a significant part of the image, the useful scene content is compressed into a narrower code-value range before enhancement is applied. Stretching this compressed histogram can expand the numerical range, but it cannot recover distinctions already lost by saturation and compression. This is one motivation for suppressing bloom before later global operations such as AEC and gamma correction.

Gamma correction occupies a different position. It is inherited from display and human-vision-oriented imaging, but a compressive nonlinearity can also benefit machine vision by reshaping the highly skewed distribution of linear RAW data. ISP4ML demonstrated that tone mapping contributes significantly to detection accuracy [21]. The important point is that machine vision needs a useful distribution transformation, not necessarily the exact gamma curve preferred for human display.

Dodge and Karam [19] demonstrated that image quality metrics designed for the HVS, including SSIM and PSNR, correlate only weakly with deep neural network recognition accuracy. Buckler et al. [20] then conducted a systematic ablation study, toggling individual ISP stages on and off across eight vision algorithms. Their key finding was that only demosaicing and gamma compression significantly affect computer vision performance, while many human-viewing-oriented stages produce little measurable benefit. Later work, including VisionISP [26] and AdaptiveISP [23], further supports the conclusion that traditional ISP modules can be aggressively pruned when the output target is machine vision.

Table 2.2 summarizes the literature findings on the effectiveness of traditional ISP modules for machine vision tasks.

Table 2.2: Literature evidence on ISP module effectiveness for machine vision.

Module	Necessity	Evidence
BLC	Necessary	Fundamental sensor calibration [27]
Demosaicing	Necessary	Pretrained models require RGB input [20]
AWB	Conditional	Corrects severe color casts [23]
AEC	Critical	Core safeguard under varying lighting [22]
Gamma	Necessary	Corrects skewed RAW distribution [20], [21]
Contrast	Omissible	Largely covered by tone mapping [23]
Saturation	Omissible	0% RL selection frequency [23], [20]
Denoising	Omissible	CNNs are relatively noise-robust [23], [20]
CLAHE	Omissible	High hardware cost and limited task-driven evidence [23]

2.2 Extreme Lighting and the Bloom Effect

Extreme lighting refers to illumination conditions that exceed the dynamic range limits of a camera sensor. In this thesis, two effects are especially relevant.

Overexposure: Intense light sources, such as the sun, vehicle headlights, and street lamps, drive sensor pixels to saturation. Once a pixel reaches the maximum RAW value, texture information in that region is lost. Pedestrians, vehicles, and traffic signs occluded by saturated pixels may become invisible.

Bloom and glare: Intense light sources can also produce large-area halos in the image. Although the light source itself may occupy only a few pixels, internal lens reflections and sensor charge overflow cause the bright region to spread outward. In Bayer color filter array (CFA) sensors, saturated values may then be further spread by demosaicing interpolation, increasing the effective area affected by the bloom.

2.2.1 Degradation of Perception Caused by Bloom

As detailed in Chapter 1, serious incidents have already occurred due to extreme-lighting degradation. This subsection therefore focuses on the image-processing mechanism behind these failures rather than repeating the accident examples.

The degradation begins with direct occlusion. Saturated pixel regions lose texture information, and objects behind the light source, such as pedestrians, vehicles, and traffic signs, may be covered by the bloom. The damage is then amplified by demosaicing interpolation. In Bayer CFA sensors, green pixels are sampled more densely than red and blue pixels and may saturate first in bright halo regions. When demosaicing reconstructs full RGB values, saturated samples can contaminate neighboring color positions, expanding the effective bloom area in the RGB image beyond its original extent in the Bayer RAW data.

The final stage is AEC global gain feedback. When bloom occupies a noticeable fraction of the frame, saturated pixels elevate the brightness statistic used by global-average metering. The AEC then reduces global gain. However, already saturated bloom regions remain saturated, while darker regions become even darker. In this

way, bloom, interpolation diffusion, and AEC feedback reinforce each other: the bloom hides local content, demosaicing expands its influence, and exposure reduction sacrifices shadow detail without removing the bloom.

2.3 Key ISP Modules

Figure 2.1 shows a concrete example of a full traditional ISP implementation, including sensor interfaces, memory access, control IPs, and many processing modules. For the module-level discussion in this section, Figure 2.2 abstracts the same architecture into two main paths: a pixel-processing data path and a sensing/control path. It should therefore be understood as a simplified explanatory view of Figure 2.1, rather than as a separate ISP pipeline.

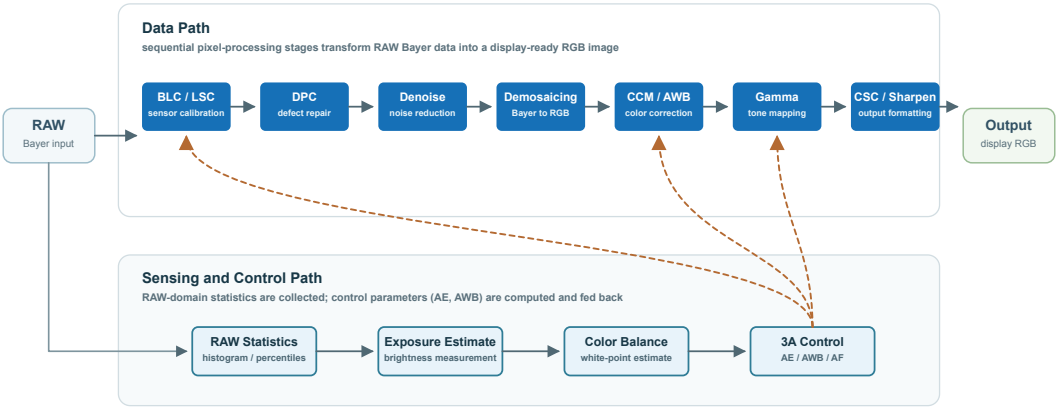

Figure 2.2: Simplified view of a traditional ISP pipeline. The data path applies a fixed sequence of pixel-processing modules, while the sensing and control path collects RAW-domain statistics and feeds back control parameters, including AE and AWB.

2.3.1 Bayer Format and Demosaicing

Each photosensitive element, or pixel, on a camera sensor can only sense luminance and is inherently incapable of distinguishing color. To capture color information, a color filter array (CFA) is deposited on the sensor surface. The most common arrangement is the Bayer pattern, a repeating 2×2 block of red (R), green (G), and blue (B) filters, with green occupying two of the four positions:

$$\begin{pmatrix} R & G \\ G & B \end{pmatrix} \quad \text{(tiled periodically across the entire sensor)}$$

Four common Bayer pattern variants, RGGB, BGGR, GRBG, and GBRG, differ only in the starting offset of the 2×2 basic block. The green proportion remains at 50%, with red and blue each at 25%.

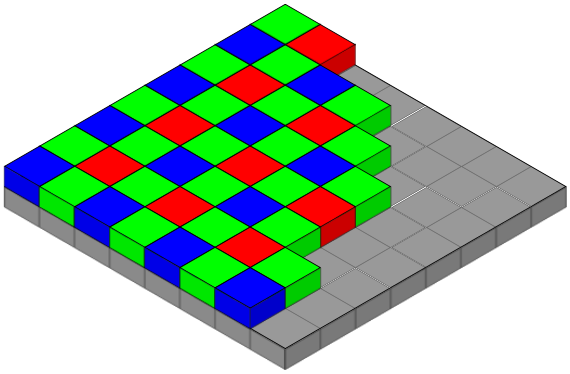

Figure 2.3: Bayer color filter array pattern on an image sensor.

The raw image produced by the Bayer CFA pattern shown in Figure 2.3 is called a Bayer RAW image: each pixel carries only one of the three color components R, G, or B. The process of reconstructing a full three-channel RGB image from the single-channel Bayer array is known as demosaicing, or CFA interpolation. Demosaicing is computationally important because it determines how single-channel sensor data becomes the RGB representation expected by most downstream perception models.

Bilinear interpolation is the most fundamental demosaicing algorithm. Missing color components at each pixel position are estimated by averaging surrounding same-color pixels. Taking the Bayer RGGB pattern as an example, at a known red pixel position $R_{i,j}$, the missing green component $G_{i,j}$ is given by the mean of its four neighboring green pixels:

$$G_{i,j} = \frac{G_{i-1,j} + G_{i+1,j} + G_{i,j-1} + G_{i,j+1}}{4}$$

For the missing blue component at the same red pixel position, the mean of four diagonal blue neighbors is used:

$$B_{i,j} = \frac{B_{i-1,j-1} + B_{i-1,j+1} + B_{i+1,j-1} + B_{i+1,j+1}}{4}$$

The same principle is applied symmetrically at blue and green pixel positions. Bilinear interpolation is simple to implement in hardware, requiring only a 3×3 window and two lines of buffering. However, it is unaware of edge direction and can spread saturated values into neighboring pixels, which is one reason bloom becomes more damaging after demosaicing.

2.3.2 BLC

BLC is a fundamental calibration operation for image sensors. Complementary metal-oxide-semiconductor (CMOS) pixels generate a weak dark-current signal even under zero-illumination conditions. Even when shielded from light, the pixel output is not exactly zero but includes a fixed offset, commonly called black level or dark-current offset. BLC subtracts this offset from each pixel's RAW value:

$$y = x - b_c$$

where b_c is the black-level offset constant for channel $c \in \{R, Gr, Gb, B\}$, typically determined by sensor factory calibration or dark-frame measurement.

Failing to perform BLC causes two problems. First, pixel values do not start from the correct zero point, causing later gain multiplication and gamma mapping to operate on a biased signal. Second, different color channels may have different black-level offsets, leading to color casts in dark regions. BLC is therefore an indispensable linear sensor-calibration module.

2.3.3 AEC

AEC dynamically adjusts the overall exposure level of an image based on scene brightness. It is a closed-loop feedback system: brightness statistics are extracted from the image, compared against a target brightness, and a global gain factor g is computed and applied to all pixels. A simple global-gain model can be written as

$$y = x \cdot g, \quad g = \frac{Y_{\text{target}}}{Y_{\text{current}}}.$$

The spatial weighting of the brightness statistics, namely the metering strategy, determines how strongly different image regions influence g. Consumer cameras typically offer multiple metering modes, including center-weighted, multi-zone evaluative, and spot metering [32]. The analysis in this thesis focuses on global-average metering, where all pixels contribute equally to the brightness statistic. This is a common baseline in embedded and FPGA-based ISP implementations because it is simple and hardware-efficient, as shown in FPGA camera prototypes such as the combined AWB/AEC design by Nilsson et al. [33].

Under global-average metering, saturated bloom regions can elevate the average brightness statistic and drive the AEC gain downward. This reduces the brightness of dark regions while leaving already saturated bloom regions largely unchanged. More advanced metering strategies can reduce this effect by down-weighting saturated areas or prioritizing regions of interest, but they also add design complexity and are outside the baseline model used in this thesis.

The central challenge of AEC is to balance response speed and temporal stability. A fast controller can track rapid lighting changes but may cause frame-to-frame oscillation, while a conservative controller is more stable but responds slowly. Temporal filtering is therefore commonly introduced. An infinite impulse response (IIR) filter smooths the per-frame gain estimate using the previous output, while a slew-rate limiter caps the maximum gain change between adjacent frames. Together, these mechanisms provide a practical trade-off between stability and responsiveness with low hardware cost.

AEC can be positioned in the ISP pipeline in two typical ways: as a processing stage within the data path, or as an independent control loop that computes gain

with frame-level feedback. The first option is simpler but consumes pixel data-path resources. The second option has almost no impact on pixel throughput but introduces a one-frame control delay. The proposed design follows the second approach, as detailed in Chapter 4.

2.3.4 Bloom Suppression

As introduced in Section 2.2, bloom is one of the main degradation mechanisms under extreme lighting. This subsection briefly revisits its physical origin only to motivate the suppression operation used in the proposed ISP pipeline.

Bloom, also referred to as halation or glare, is produced when light sources exceed the optical and electrical limits of the imaging system. Intense light can undergo internal reflections between lens elements, forming a diffuse halo around the source. In addition, excess charge from saturated sensor pixels may spread to neighboring pixels. In Bayer CFA sensors, demosaicing can further expand the affected region by interpolating saturated values into neighboring color positions.

The core idea of bloom suppression is to apply localized gain attenuation in the halo region, rather than relying only on global AEC gain reduction. The general processing pipeline consists of four steps, as summarized in Table 2.3.

Table 2.3: Four-step general pipeline for bloom suppression.

Step	Description
1	Estimate local brightness via box filter
2	Identify halo by comparing local brightness against pixel value
3	Apply suppression gain (<1.0) to halo only; gain = 1.0 elsewhere
4	Combine local bloom gain with global AEC gain in one multiplication

The idea of comparing a pixel with its local neighborhood is related to Retinex theory, which states that perceived lightness depends on local relationships rather than absolute intensity alone [7]. Local tone mapping later applied similar neighborhood comparisons for display-oriented enhancement [34]. In this thesis, the same general principle is adapted for a different purpose: reducing bloom influence before demosaicing and before downstream machine-vision processing. The architectural implications of this Bayer-domain placement are discussed in Chapter 6.

2.3.5 AWB

AWB aims to reduce global color shifts caused by the illumination source and the sensor response. Different light sources, such as daylight, incandescent light, and fluorescent light, have different spectral power distributions, causing the same object to appear with different RGB values. AWB estimates the scene's illumination color temperature and applies independent gain corrections to each color channel:

$$R' = R \cdot g_R, \quad G' = G \cdot g_G, \quad B' = B \cdot g_B.$$

Typically, the green channel serves as reference ($g_G = 1.0$), with only the red and blue channel gain ratios adjusted.

The necessity of AWB also arises from the physical characteristics of CMOS sensors. In the Bayer CFA arrangement, green pixels occupy 50% of the array while red and blue each occupy 25%. In addition, silicon photodiodes have different conversion efficiencies at different wavelengths. The objective of AWB is therefore to compensate for these channel-response differences and restore a neutral balance under a given light source.

Gray World is one of the classic AWB algorithms. Its assumption is that the average color of all objects in a scene should be neutral gray, meaning that the R, G, and B channel means are equal. The correction gains for the red and blue channels are then computed from this assumption. For applications with fixed camera modules, gain parameters can also be determined through one-time calibration, reducing the need for per-frame dynamic estimation.

2.3.6 Gamma Correction

Gamma correction is a fundamental step in the digital image processing pipeline. It applies a nonlinear mapping to image data before display or later processing.

Historically, gamma correction was introduced to compensate for the nonlinear response of display devices. The screen luminance L of a cathode ray tube (CRT) display follows an approximate power-law relationship with the input drive voltage V: $L = V^{\gamma}$, where γ is approximately 2.2–2.5 [29]. Although modern displays no longer rely on the same physical mechanism, the $\gamma \approx 2.2$ convention remains common for compatibility.

The corresponding camera-side encoding can be written as

$$y = 255 \times \left(\frac{x}{255}\right)^{1/\gamma}.$$

This mapping allocates more code values to darker regions and fewer to bright regions. This is useful for human viewing because human brightness perception is nonlinear [29]. It is also useful for machine vision because linear RAW data often has a highly skewed distribution, with dark-region information densely packed into a narrow numerical range.

Figure 2.4 shows the transfer curves for several γ values. Higher γ values apply stronger nonlinear compression and allocate more output resolution to dark regions, while lower γ values preserve more separation in midtone and highlight regions. For downstream neural networks, the optimal mapping is not necessarily the same as the display-oriented standard. The useful property is the redistribution of code values into a form better matched to recognition models.

Because the power function compresses some gray-level intervals, gamma correction is a lossy tone-mapping operation. Hansen et al. [21] demonstrated through systematic ablation experiments that tone mapping independently contributes over 5

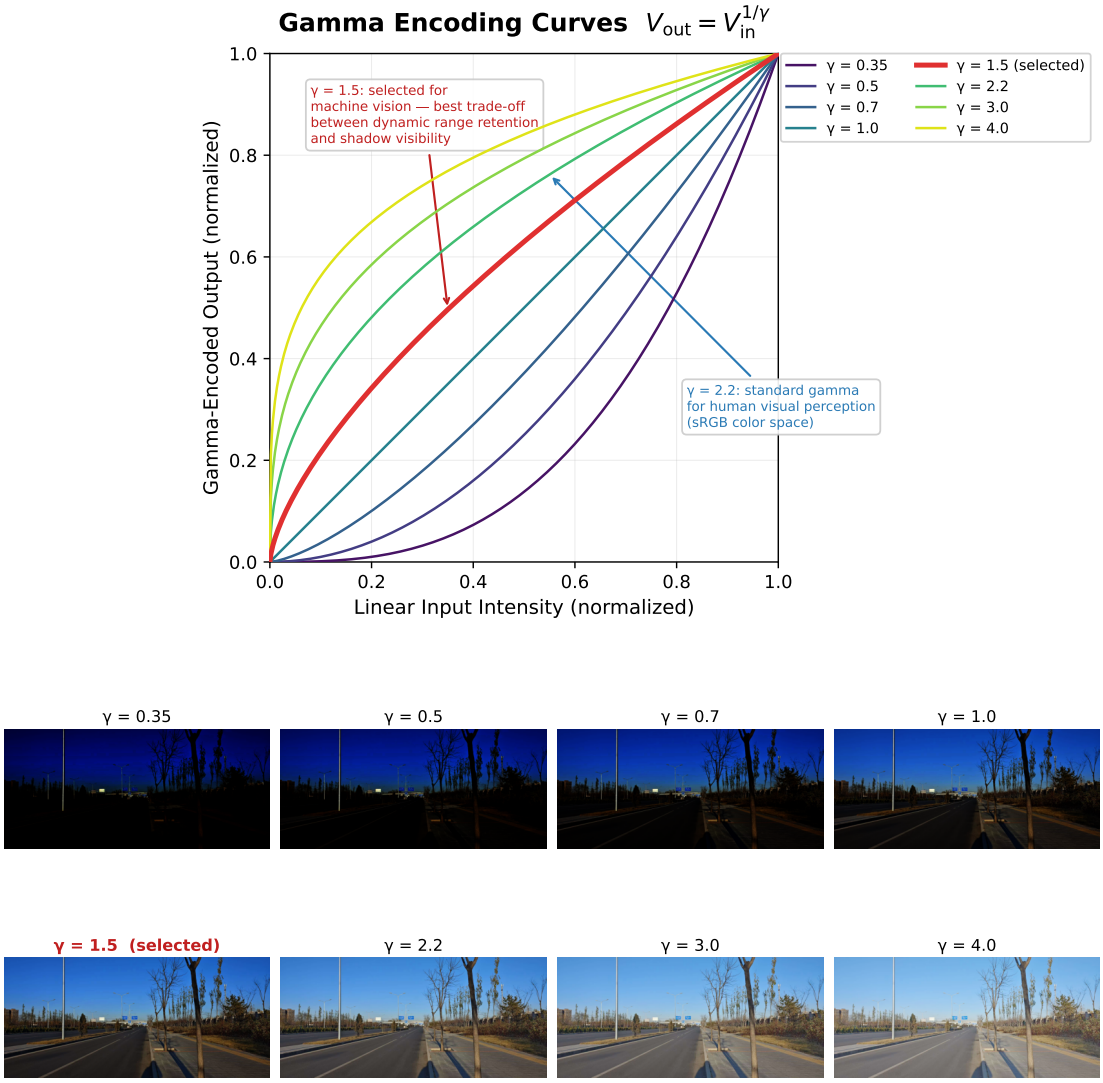

Figure 2.4: Gamma correction transfer curves for different γ values.

percentage points of detection accuracy. The value of gamma correction for downstream vision tasks lies in transforming the skewed distribution of linear RAW data into a more useful distribution for neural network inference.

2.4 Hardware Platform

2.4.1 FPGA Device: Xilinx xc7z035

The selected device is the Xilinx Zynq-7000 series `xc7z035ffg676-2` (Speed Grade -2). This device belongs to the Zynq SoC family, integrating a dual-core ARM Cortex-A9 processor on the processing system (PS) side with programmable logic (PL), making it suitable for hardware-software co-design in embedded vision processing.

Table 2.4: xc7z035ffg676-2 resource summary.

Resource	Quantity
Logic Cells	275,000
Slice LUTs	171,900
Slice Registers	343,800
DSP Slices	900
Block RAM	500 tiles (17.6 Mb)

The selected development board is the F29_CZ10_7035 (Figure 2.5), which provides a MIPI CSI-2 camera interface and HDMI video output interface, along with driver reference designs and MIPI IP core configuration solutions. These features accelerate the development of video processing systems. The resource summary of the xc7z035 device is given in Table 2.4; its BRAM and DSP resources provide hardware headroom for implementing the proposed image processing pipeline.

Figure 2.5: F29_CZ10_7035 development board with Xilinx Zynq xc7z035.

2.4.2 Image Acquisition Devices

The camera module used in this system features a Sony IMX291 sensor (Figure 2.6), a 1/2.8" CMOS device with low-light sensitivity that outputs 1920×1080 @ 30 fps Bayer RAW data via a MIPI CSI-2 interface. This sensor offers a high signal-to-noise ratio under low-light conditions, making it suitable for nighttime driving scenario testing.

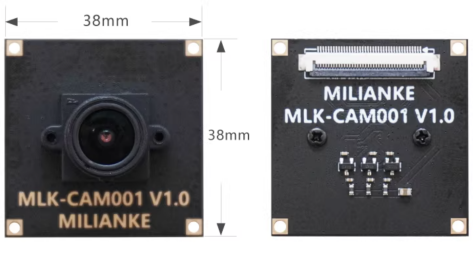

Figure 2.6: Sony IMX291 camera module with MIPI CSI-2 interface.

As an image-quality reference benchmark, the DJI Action 5 Pro (Figure 2.7) was used for synchronized comparison capture. It provides a high-quality consumer ISP pipeline and can record 1080p video at 30 fps for comparison. In this thesis, it is used only as an image-quality reference, not as a candidate system for autonomous-driving deployment.

Figure 2.7: DJI Action 5 Pro used as image-quality reference benchmark.

2.4.3 Video Capture Card

A video capture card converts external video signals, such as HDMI and SDI, into a data stream recognizable by a computer, typically via USB or PCIe. In this system, the FPGA outputs the processed video signal via HDMI; the video capture card (Figure 2.8) converts the HDMI signal to a USB video stream; and the computer performs real-time recording and monitoring through OBS software.

Figure 2.8: Video capture card used for HDMI-to-USB conversion.

2.5 Validation Tools and Datasets

2.5.1 CUT Model: Unpaired Image-to-Image Translation

In this work, we need to evaluate whether images processed by our ISP pipeline harm downstream model accuracy. This requires test images that have both our camera's imaging style and high-quality annotations. Since we cannot physically recapture existing annotated datasets with our camera, we instead use a computational approach: a model that can take any image and make it look as if it were captured through our ISP pipeline, without changing the scene content or annotations.

CUT (Contrastive Unpaired Translation) [35] achieves this by learning the visual difference between two sets of images without requiring matched pairs of the same scene. The model transforms an image from one domain so that it looks like it belongs in the other, while preserving the content structure of the original image. This makes it suitable for generating style-transferred images as a proxy for real captures.

A critical property of CUT is that it is a lossy mapping: it can recombine and transform existing visual information, but it cannot add new valid scene content. This property is central to the validation strategy, because comparable model accuracy between original and CUT-transferred data indicates that the ISP-induced style change does not substantially harm downstream models. Figure 2.9 shows an example of the style transfer effect.

As shown in Figure 2.9, style-transferred images preserve the original scene content and semantics while adopting the imaging characteristics of the target camera, including color response, noise distribution, and tone mapping.

2.5.2 Evaluation Models

To verify whether ISP processing affects downstream recognition tasks, controlled experiments are conducted on standard benchmark datasets using representative models. Table 2.5 summarizes the selected models.

YOLOv8m is a single-stage object detection model from the YOLO series [36]. It is used in this thesis because real-time detection of vehicles and pedestrians is directly

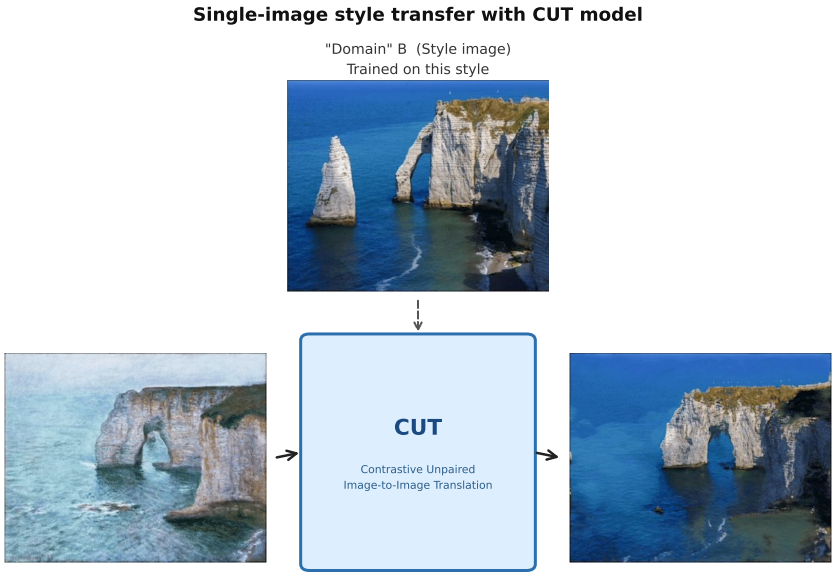

Figure 2.9: CUT style transfer effect.

Table 2.5: Evaluation models used for ISP processing validation.

Model	Task	Dataset
YOLOv8m	Object Detection	KITTI
SegFormer-B0	Semantic Segmentation	Cityscapes

relevant to autonomous driving perception. Its high-level semantic feature extraction also makes it suitable for testing whether ISP-induced style changes harm object detection performance.

SegFormer-B0 [37] is a Transformer-based semantic segmentation model. As a pixel-wise dense prediction task, semantic segmentation imposes higher requirements on local feature consistency than object detection, making it useful for evaluating whether ISP-induced style variations affect per-pixel prediction.

2.5.3 Benchmark Datasets

Table 2.6: Benchmark datasets used for evaluation.

Dataset	Task
KITTI [38]	Object detection over urban, rural, and highway driving scenes
Cityscapes [39]	Semantic segmentation across urban street scenes with fine annotations

The two benchmark datasets used in the experiments are summarized in Table 2.6. The KITTI dataset [38] is one of the most influential benchmark datasets in autonomous driving. It contains driving scenes recorded with vehicle-mounted sensors in Karlsruhe, Germany, and is used here for object detection evaluation. Figure 2.10 shows a representative scene from the dataset.

Figure 2.10: KITTI dataset example scene.

The Cityscapes dataset [39] is designed for urban scene semantic understanding. It contains high-resolution street images with fine semantic annotations across 19 classes, making it suitable for segmentation-based validation. Figure 2.11 shows example street scenes from the dataset.

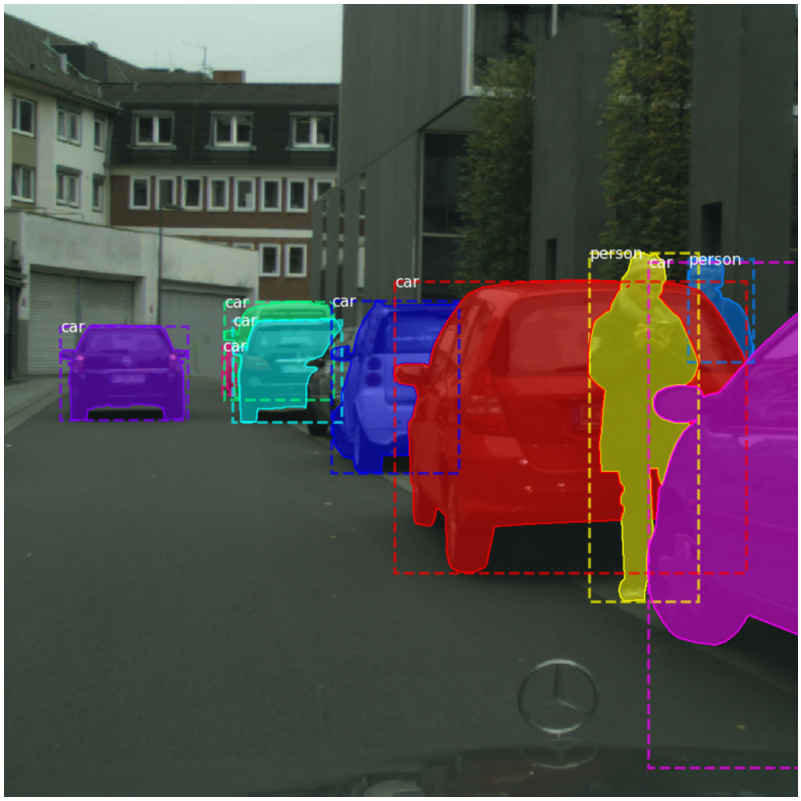

Figure 2.11: Cityscapes dataset street scenes.

3

Method

Building on the technical background in Chapter 2, this chapter describes the design and validation methodologies used to develop and evaluate the proposed ISP system. Section 3.1 outlines the two-step simplification approach and the simplicity-first design philosophy. Section 3.2 covers the validation strategy, including the hardware verification workflow and the style-transfer-based downstream effectiveness validation.

3.1 Design Methodology

The core objective of this work is to achieve high processing speed with low and deterministic latency. To this end, the system design follows a two-step simplification approach. First, processing modules that provide no measurable benefit to downstream machine vision tasks are removed based on the literature consensus established in Chapter 2. Second, the retained modules are decomposed into atomic operation types. Isomorphic repetitions are then identified, and adjacent operations of the same type are combined through function composition, reducing multiple pipeline stages to the minimum number of sequential passes.

A simplicity-first design philosophy guides the entire process. When a simple solution satisfies the design specifications, a more complex alternative is avoided unless it provides a clear validation or performance benefit. The resulting hardware architecture is described in Chapter 4.

3.2 Validation Methodology

Validation proceeds at two levels: hardware correctness and downstream effectiveness.

3.2.1 Hardware Verification

The hardware verification workflow employs three components in a progressive chain, moving from rapid algorithm exploration to cycle-level hardware checking.

A Python implementation of the core algorithms serves as the earliest validation stage. Operating on full-frame images without timing constraints, it enables rapid prototyping and visual inspection of algorithmic behavior. This stage confirms that

the image processing logic produces the intended results before hardware constraints are considered.

A C++ reference model then replaces the Python prototype. While functionally identical in its image processing logic, the C++ implementation runs with substantially higher efficiency, making it feasible to process large volumes of video footage and examine the output across diverse scenes. This provides a faster, more scalable, and higher-confidence validation of the algorithms than the Python stage alone.

After the algorithm has been validated in software, the register-transfer level (RTL) hardware description is written following the same design. Finally, a Verilator-compiled RTL simulation [40] closes the verification loop. Verilator compiles the synthesizable RTL design into a C++ simulation model, allowing the same C++ testbench that drives the reference model to also drive the RTL under identical per-cycle stimulus. The outputs of the C++ reference model and the RTL simulation are compared pixel by pixel in code, and a zero-mismatch result is required for the test to pass. This architecture, first demonstrated in the FPGA ISP domain by Darkroom [27], is illustrated in Figure 3.1.

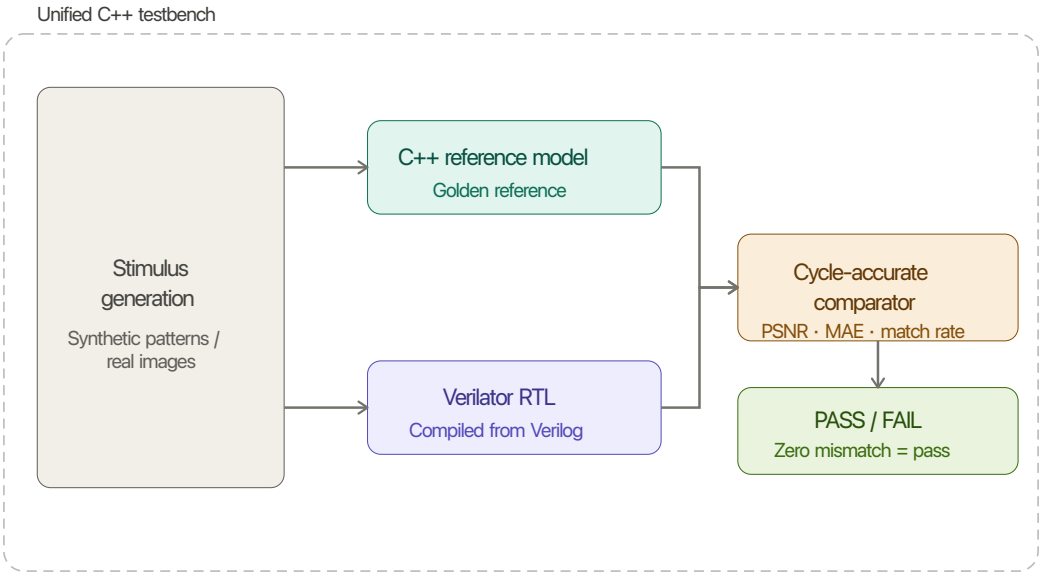

Figure 3.1: Golden-reference co-verification architecture.

Verification proceeds at three levels. At the module level, each ISP stage is verified independently using synthetic test patterns covering the complete input space. At the pipeline level, all modules are connected in series, verifying cross-module interface timing and control signal alignment. At the image level, the reference model runs on real photographic images, enabling rapid parameter-tuning iteration without RTL simulation. All modules share a unified streaming interface with control signals delayed through shift registers matching the data pipeline depth. Automated regression with containerized environments ensures cross-platform reproducibility.

3.2.2 Downstream Effectiveness Validation

This system targets autonomous driving scenarios, where the downstream components are machine vision perception models, including object detection and semantic segmentation. The central validation question is whether the system's ISP processing adversely affects the recognition accuracy and training behavior of these downstream models. The basic validation strategy is to train models on data bearing the system's imaging style and compare the training dynamics and final accuracy against models trained on the original data.

Directly capturing and annotating a dedicated dataset is physically impractical. High-quality per-pixel annotation is prohibitively expensive, and a self-annotated dataset cannot match the scene diversity and annotation consistency of publicly available benchmarks. This work therefore adopts contrastive unpaired translation (CUT) [35] as the validation vehicle. CUT learns the imaging style differences between road scenes captured through the system's pipeline and existing public benchmark datasets, then transfers the learned style onto the benchmarks. The transferred images retain the original scene content and per-pixel annotations, while only the visual appearance is changed to resemble the system's ISP output.

The experimental design follows a single-variable control principle: all training configurations are held constant, and only the imaging style of the training data is varied. Models are trained and evaluated on two representative perception tasks, object detection and semantic segmentation, using publicly available benchmark datasets. Since CUT is a lossy mapping, it supports a conservative lower-bound interpretation. If model accuracy on transferred data matches that on original data, the system's ISP-induced style variations at least do not harm downstream models. The scientific rationale for choosing style transfer as the validation strategy and a detailed discussion of the lower-bound interpretation are deferred to Chapter 6.

3.2.3 Functional Validation

In parallel, a qualitative functional validation is conducted. The system output and a professional reference camera record synchronized video of the same road scenes under extreme lighting conditions. The two outputs are compared side by side through visual inspection. If the system achieves comparable suppression of glare and bloom artifacts while preserving dark-region detail relative to the reference, functional adequacy is confirmed. The corresponding visual comparison results are presented in Chapter 5.

4

System Design

Building on the module analysis in Chapter 2 and the methodology in Chapter 3, this chapter presents the hardware architecture, physical setup, and downstream validation configuration of the proposed FPGA ISP system. The chapter first explains how the pruned ISP modules are compressed into a three-stage streaming pipeline with an AEC side channel. It then describes the acquisition setup and the validation experiment configuration.

4.1 Pipeline Design Rationale

The core per-pixel task of this system can be stated simply: determine, for each Bayer RAW pixel, an appropriate gain value such that underexposed pixels are lifted, pixels inside bloom halos are suppressed, and normal pixels pass through unchanged. This per-pixel gain is the product of two components. The global component g_{global} derives from frame-wide brightness statistics. When the scene as a whole is dark, the global gain rises; when it is bright, the gain falls. The local component g_{local} derives from an 11×11 neighborhood centered on the pixel. If the neighborhood is substantially brighter than the pixel itself, indicating a bloom halo, the local gain is suppressed to $0.5\times$; otherwise it remains at unity. The two components are computed independently: the global gain is produced once per frame by the AEC side channel, and the local gain is produced per pixel by the bloom-detection logic in Stage 1. They are then composed into g_{combined} and applied in a single multiply-add operation.

The system architecture follows a single design objective: maximize processing throughput while minimizing per-pixel latency. Following the two-step simplification methodology established in Chapter 3, a conventional pipeline of over ten sequential stages is reduced to the minimum number of passes.

Step one is module pruning. Based on the literature consensus established in Chapter 2, modules that mainly serve human-viewing quality and offer no clear benefit to convolutional networks, including sharpening, denoising, saturation enhancement, contrast stretching, and CLAHE, are removed. After pruning, only five essential modules remain, as listed in Table 4.1: BLC for fundamental sensor calibration, AEC as the safeguard under varying lighting, demosaicing to satisfy the RGB input requirement of pretrained models, AWB to correct extreme color casts, and gamma to reshape the skewed RAW distribution for stable gradient optimization.

Table 4.1: Modules retained in the pruned pipeline.

Module	Rationale
BLC	Fundamental sensor calibration
Demosaicing	Pretrained models require 3-channel RGB input [20]
AEC	Core safeguard under varying and extreme lighting [22]
Gamma	Corrects skewed RAW distribution for stable gradients [21]
AWB	Only extreme color casts must be corrected; fixed gains suffice [23]

Step two is mathematical fusion. The five retained modules are not independent: they can be decomposed into lower-level operation types. BLC uses offset subtraction, AWB and AEC both use linear gain multiplication, gamma uses nonlinear LUT mapping, and demosaicing together with the bloom box filter both use spatial convolution. These operations fall into two classes with different simplification potential. Pointwise transforms, including BLC, AWB, AEC, and gamma, depend only on the same-position input pixel and can be composed. Spatial transforms, including demosaicing and the bloom box filter, depend on neighboring pixels and therefore require dedicated buffering. Accordingly, Stage 1 combines BLC offset, AEC gain, and local bloom suppression into a single Bayer-domain affine operation, while Stage 3 pre-combines AWB gain and gamma into one set of LUTs. The module compression is shown in Figure 4.1.

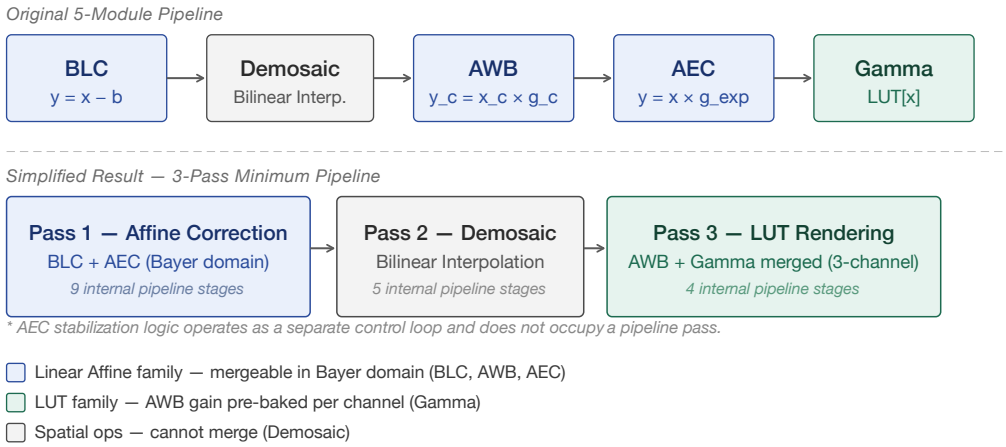

Figure 4.1: Module compression analysis.

The final result is a three-stage pipeline with one side channel, containing 9, 5, and 4 internal pipeline stages respectively. Stage 1 performs BLC, AEC gain, and local bloom suppression together in the Bayer domain; Stage 2 performs demosaicing; Stage 3 applies AWB and gamma correction simultaneously through pre-combined LUTs; and the AEC gain computation executes asynchronously in an independent side channel, contributing zero latency to pixel throughput.

4.1.1 Overall Architecture

The complete system architecture is shown in Figure 4.2. The pixel data path consists of a three-stage streaming pipeline operating on 4 pixels per clock cycle: Bayer RAW data enters at 40 bits (4 pixels × 10 bits) and exits as 24-bit RGB. In parallel, an AEC side channel monitors the RAW input, computes gain once per frame during the vertical blanking interval, and feeds the result back to Stage 1. This forms a closed perception-control loop without stalling the pixel stream. Because Figure 4.2 is a high-level architecture diagram, its internal labels should be read together with the following clarification: Stage 2 denotes only the bilinear demosaicing stage, while fixed Gray World/AWB calibration is implemented in Stage 3 through the combined AWB–gamma LUT. The AEC side channel uses a five-state gain-decision FSM.

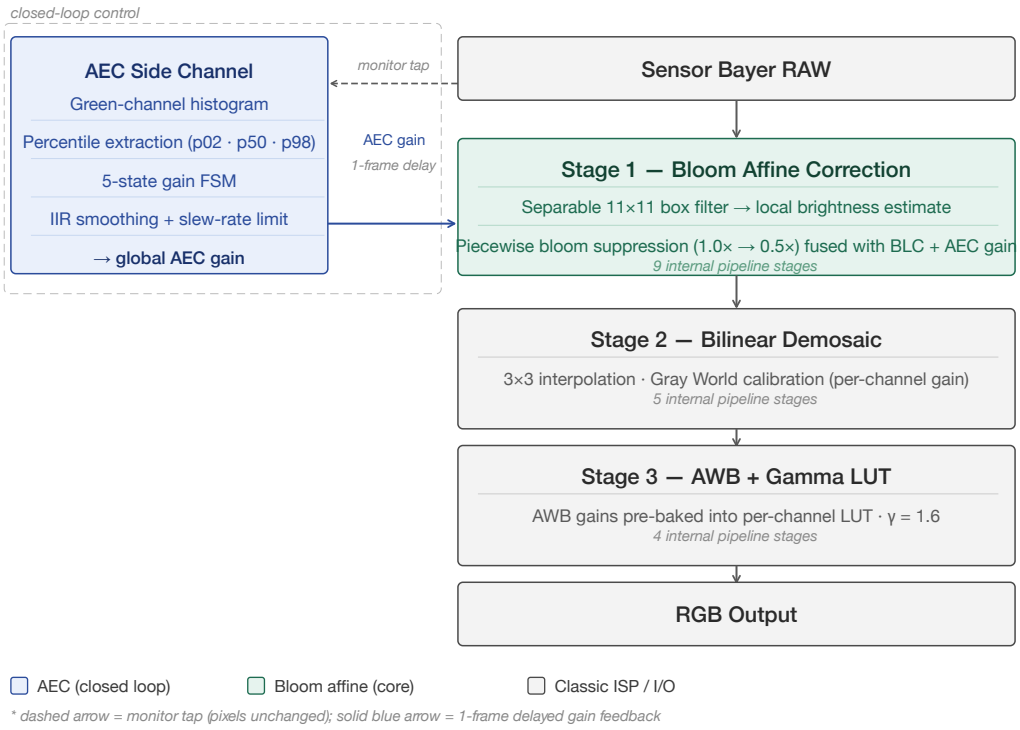

Figure 4.2: Three-stage pipeline architecture with AEC side channel. Stage 2 performs bilinear demosaicing only, while fixed AWB calibration is implemented in Stage 3 through the combined AWB–gamma LUT. The AEC side channel uses a five-state gain-decision FSM.

The following subsections begin with the AEC side channel, which supplies the global gain and scene statistics that the downstream pipeline stages depend on, before detailing each of the three pixel-data-path stages in sequence.

4.1.2 AEC: Global Gain Control

The task of the AEC module is to supply the entire system with a frame-level global gain g_{global}, a single scalar value bounded between 0.25× and 1.5× that determines

the overall exposure level of the next frame. If the scene as a whole is dark, g_{global} rises to brighten the output; if strong light sources occupy a large fraction of the frame and inflate the brightness statistics, g_{global} falls to prevent overexposure. Architecturally, the AEC is implemented as a side channel independent of the pixel data path. It passively monitors the RAW input stream, collects statistics, computes the gain during the inter-frame vertical blanking interval, and feeds the result back to Stage 1.

The implementation principle is as follows: collect a 256-bin luminance histogram frame by frame; during vertical blanking, extract the p02, p50, and p98 percentiles; feed them to a five-state priority-ordered FSM that determines the gain strategy; and pass the raw decision through dual-layer temporal filtering, namely an IIR low-pass filter followed by a slew-rate limiter, before output. The IIR filter stores only the smoothed gain from the previous frame, so inter-frame awareness is achieved at the hardware cost of a single register. Figure 4.3 illustrates the complete FSM architecture.

The AEC gain computation is separated from the pixel data path because the gain is a frame-level quantity that changes at most once per frame, while its computation requires global statistics available only after the entire frame has been scanned. Inserting this computation into the per-pixel streaming path would either stall the pipeline or require complex scheduling.

The side channel monitors the Bayer RAW input stream at the pipeline entry point through zero-latency pass-through tapping. During active frame scanning, it performs two concurrent operations. First, it identifies green pixels according to the configured Bayer pattern and increments the corresponding bin in a 256-bin histogram, organized as 4 banks to match the 4-pixel-per-cycle processing rate. Second, it maintains overexposure and underexposure pixel counters that tally pixels exceeding or falling below configurable thresholds, providing direct saturation-zone and noise-floor awareness independent of the histogram shape.

At each frame boundary, during the vertical blanking interval when no pixel data flows, the AEC state machine transitions from monitoring to computation. It sequentially scans the 4 histogram banks, accumulating them into a single cumulative distribution, and latches three key percentiles: p02, representing the shadow floor; p50, representing the overall exposure level; and p98, representing the highlight ceiling. A sequential divider then computes a highlight-safe gain ceiling from p98 and a shadow-lift gain from p02.

The computed percentiles and pixel counters feed a five-state FSM that determines the gain strategy for the next frame. The decision policy, shown in Table 4.2, is ordered by descending priority. Higher-priority conditions are evaluated first and preempt lower ones.

Raw gain decisions from the FSM are passed through an IIR low-pass filter that smooths gain changes across frames, followed by a slew-rate limiter enforcing a hard cap of approximately $\pm 3\%$ of the full gain range on per-frame gain steps. This dual-layer temporal filtering is essential for video applications, where inter-frame brightness oscillation is more detrimental to object tracking and feature matching

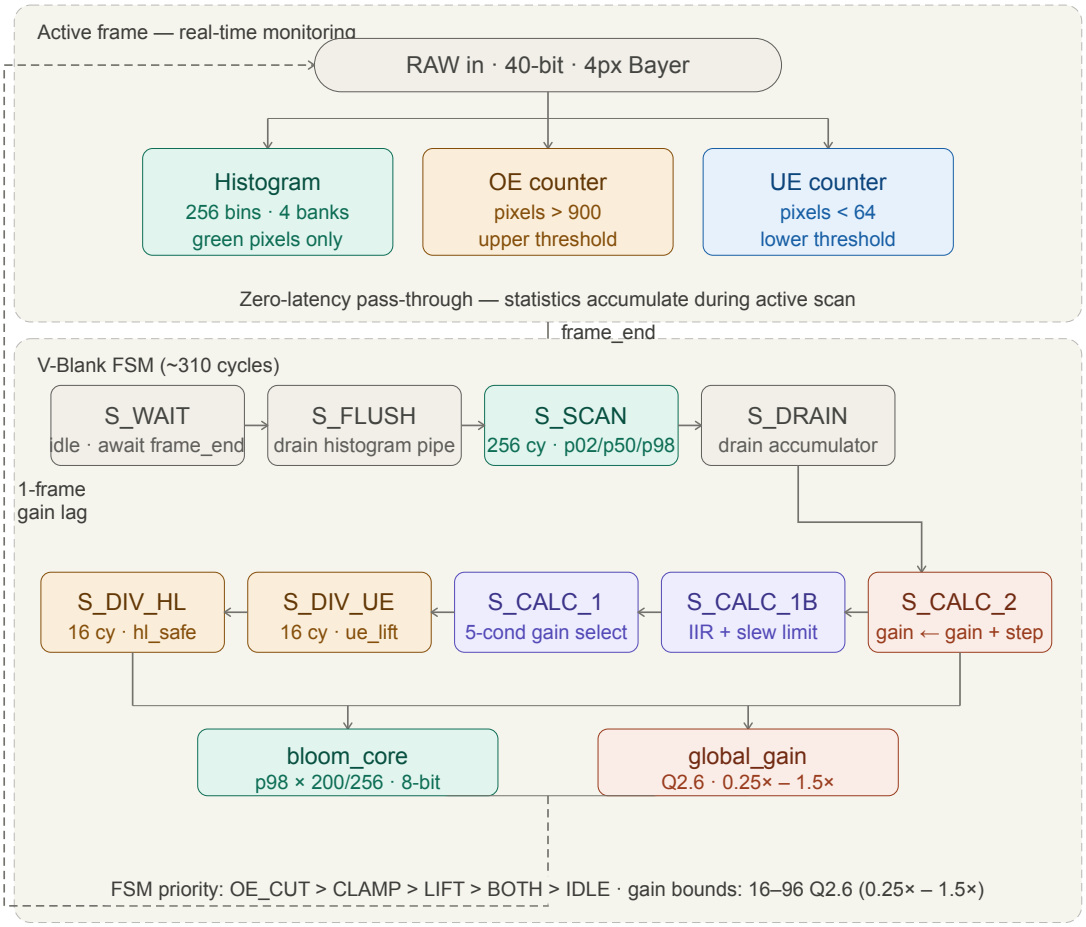

Figure 4.3: AEC constraint FSM: histogram collection and five-state V-Blank gain computation.

Table 4.2: AEC decision policy.

State	Condition	Action
Emergency Cut	Overexposed pixel count exceeds 1/128 of total pixels	Immediate gain reduction
Clamp	p98 exceeds upper safety threshold	Reduce gain to highlight-safe ceiling
Lift	p02 at or below noise floor	Increase gain to lift shadows
Both	Both clamp and lift conditions active	Prioritize clamp
Idle	All percentiles within safe operating range	Maintain current gain

than a slight delay in reaching the optimal exposure. The overall gain range is bounded between hard limits of 0.25× and 1.5×, preventing extreme under- or over-exposure regimes from which recovery would be slow.

4.1.3 Stage 1: Bloom Affine

The core responsibility of the first pipeline stage is to compute a per-pixel local gain g_{local}, multiply it with the global gain g_{global} supplied by the AEC module into a single g_{combined}, and apply both together with the BLC dark-current offset to each pixel in one multiply-add operation. This stage implements the architectural idea motivated in Chapters 1 and 2: gains and offsets are applied in the Bayer RAW domain before demosaicing, so bloom is suppressed before saturated energy is spread by interpolation.

The implementation principle is as follows: a 9-stage pipeline first estimates local neighborhood brightness via a separable 11×11 box filter, determines whether each pixel lies inside a bloom halo by comparing neighborhood brightness against the pixel's own value, normalizes this comparison against a scene-adaptive threshold derived from the AEC p98 statistic, converts the result to a local gain in the 0.5–1.0× range through linear interpolation, and finally multiplies it with the global gain and applies the result to the BLC-corrected RAW pixel. Figure 4.5 shows the complete pipeline structure.

Bloom suppression is implemented as a two-step process: locate strong light-source neighborhoods, then suppress the halo around them. The system estimates the average brightness within an 11×11 window centered on the pixel and compares this neighborhood average against the pixel's own value. If the neighborhood is substantially brighter than the pixel itself, the pixel is classified as lying inside a bloom halo. The attenuation is continuous rather than binary, following the gain curve in Figure 4.4. When the normalized excess is zero, the gain remains at 1.0×. As the excess increases, the gain descends linearly toward 0.5×. This suppresses the halo enough to recover texture behind it without turning the region black.

The choice of an 11×11 window reflects a balance of detection range and hardware cost. At 1080p resolution, point light sources such as vehicle headlamps and streetlights produce bloom transition zones on the order of tens of pixels. A radius of 5 pixels allows the window to extend from the halo periphery toward the bright core, producing a useful neighborhood-vs.-pixel brightness difference. Larger windows add line buffers and arithmetic cost with limited additional benefit. In addition, the division by 121 can be approximated efficiently as $\text{vsum} \times 542 + 32768 \gg 16$, avoiding a true divider. The trade-off is summarized in Table 4.3.

These nine stages partition naturally into three logical phases. The first four stages, S0–S3, constitute the spatial filtering phase: they compute the 11×11 neighborhood-average brightness for each pixel. The middle three stages, S4A–S4C, constitute the gain-decision phase: they compare neighborhood brightness against the pixel's own value, quantify the degree of bloom, and map the result to a local gain in the 0.5–1.0× range. The final two stages, S5A–S5B, constitute the gain-application phase: they multiply the local gain with the AEC global gain and apply both, together

Table 4.3: Window size trade-off for bloom box filter.

Radius	Window	Added Line Buffers	Detection Range Gain
3	7×7	baseline	insufficient for wide halos
5	11×11	+4	adequate for typical halos
7	15×15	+4 more	limited further gain

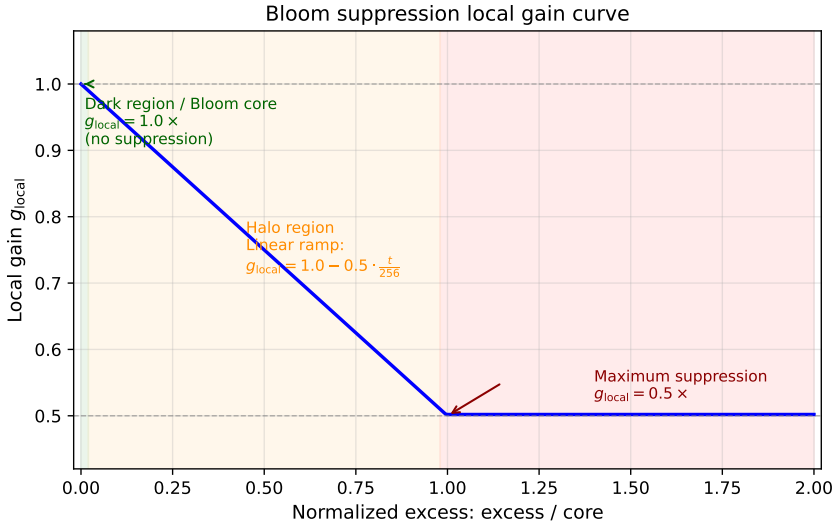

Figure 4.4: Bloom local gain curve. Horizontal axis: normalized excess (excess / core). Vertical axis: local gain g_{local}.

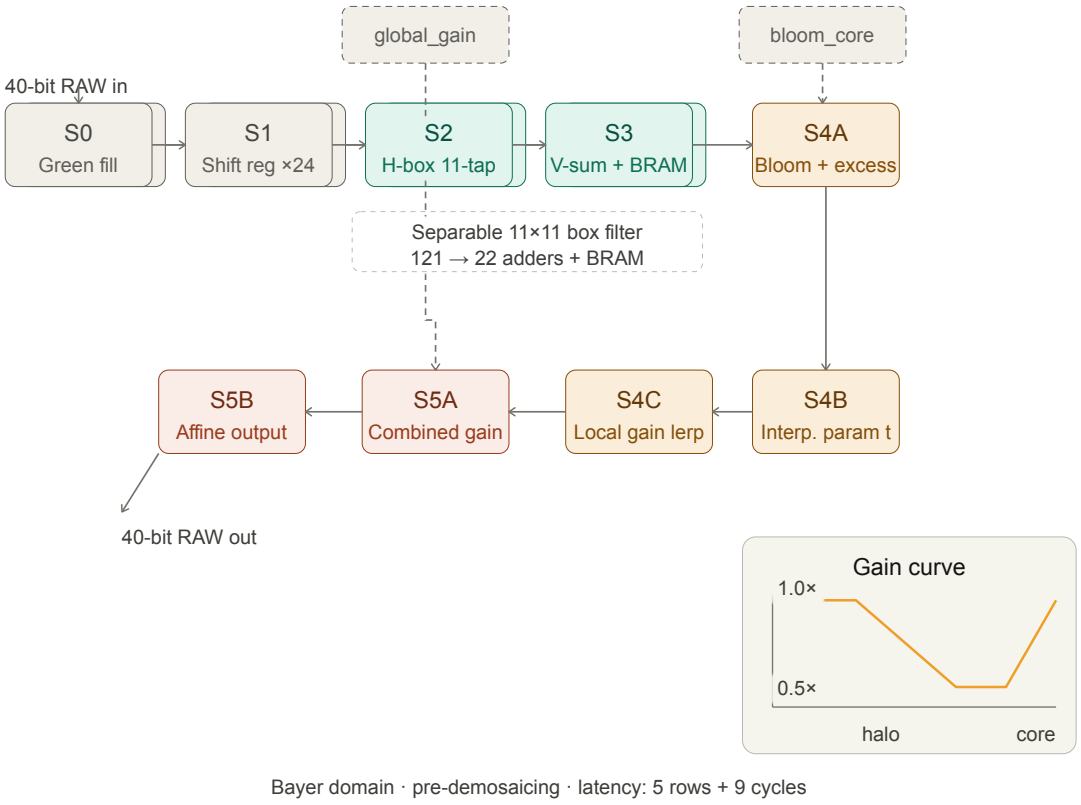

Figure 4.5: Nine-stage bloom affine correction pipeline.

with the BLC offset, in a single multiply-add operation.

S0: Green fill. The box filter that detects bloom halos needs a brightness value at every pixel, but the Bayer sensor provides green at only half the positions. S0 fills the gaps by borrowing the nearest horizontal green value from the same 4-pixel beat, producing a full-resolution luminance estimate on a separate path that leaves the main RAW data unchanged.

S1: Shift register. Computing an 11-wide sliding window sum requires simultaneous access to 11 consecutive pixel values, while the pipeline receives only 4 new pixels per cycle. S1 buffers 24 entries of history in a shift register, giving the next stage enough temporal context to form the full 11-tap window centered at each output position.

S2: Horizontal box sum. The 11×11 box filter is implemented as a separable filter. S2 performs the horizontal pass, computing the sum of 11 consecutive green-filled pixels in a sliding window and producing four horizontal sums per cycle, one per output pixel.

S3: Vertical sum and RAW delay. S3 completes the box filter by summing 11 rows of horizontal partial sums into a single vertical sum. At the same time, it reads the original raw10 pixel from a delay buffer so that the main pixel data is aligned with the box-filter output before the gain-decision phase.

At the close of the spatial filtering phase, each pixel has acquired the two values needed for bloom judgment: the total brightness vsum of its 11×11 neighborhood, and its own RAW value raw10. The gain-decision phase then converts these two numbers into a local gain.

S4A: Bloom brightness and excess. The box-filter sum is first scaled to an 8-bit bloom brightness estimate by a DSP multiply-shift that approximates division by 121 without a true divider, then compared against the pixel's own value:

$$\begin{aligned} \text{bloom} &= \frac{\text{vsum} \times 542 + 32768}{2^{16}} \\ \text{raw8} &= \text{raw10} \gg 2 \\ \text{excess} &= \max(0,\ \text{bloom} - \text{raw8}) \end{aligned}$$

A positive excess means the neighborhood is brighter than the pixel; the pixel sits inside a bloom halo. A zero excess means the pixel is not in a halo.

S4B: Interpolation parameter. The raw excess is normalized against a scene-adaptive core threshold supplied by the AEC side channel, replacing an expensive true division with a ROM lookup and DSP multiplication:

$$\begin{aligned} \text{core} &= \frac{\text{p98} \times 200}{256} \\ \text{inv\_core} &= \frac{65536}{\text{core}} \\ t &= \frac{\text{excess} \times \text{inv\_core}}{2^8} \end{aligned}$$

Because core tracks the scene's 98th-percentile brightness, the normalization adapts automatically.

S4C: Local gain interpolation. With the normalized parameter t, S4C converts the bloom estimate into a gain value by linear interpolation between unity and the minimum gain:

$$g_{\text{dark}} = 1.0\times$$
$$g_{\text{halo}} = 0.5\times$$
$$g_{\text{local}} = g_{\text{dark}} + \frac{(g_{\text{halo}} - g_{\text{dark}}) \times t}{2^8}$$

The result implements the four-region model in Table 4.4 using a single DSP multiplication and a shift.

S5A: Gain composition. Two independent gain factors now exist: the global AEC gain and the local bloom gain. S5A multiplies them first into a single coefficient, so that only one rounding operation is needed per pixel:

$$g_{\text{combined}} = \frac{g_{\text{global}} \times g_{\text{local}}}{2^6}.$$

S5B: Affine output. The raw10 value is corrected by the black-level offset and multiplied by the combined gain in a single operation:

$$\text{out} = \text{clamp}\Big(\frac{(\text{raw10} - b) \times g_{\text{combined}}}{2^6},\ 0,\ 1023\Big).$$

With the black-level offset b at its default of zero, the output is a corrected Bayer RAW pixel in which global exposure, local bloom suppression, and sensor calibration have all been applied simultaneously.

Table 4.4: Excess gain model.

Region	Condition	Gain
Bloom core	RAW value above core threshold	1.0×
Dark region	Excess = 0	1.0×
Halo region	0 < excess < core threshold	Linear interpolation from 1.0× to 0.5×
Maximum suppression	Excess above core threshold	0.5×

The global AEC gain and local bloom gain are both represented in Q2.6 fixed-point format. They are multiplied into a single Q2.6 coefficient in Stage 5A and applied together with the BLC offset in a single affine operation per pixel. This single-multiplication approach avoids the accumulated rounding error of cascaded per-pixel multiplications.

The critical design decision is applying all gains and offsets in the Bayer domain, before demosaicing. Linear gain scaling is compatible with interpolation, so Bayer-domain application preserves the effect of the corresponding RGB-domain gain while

acting before saturated energy diffuses through demosaicing. This timing is the key advantage: bloom is suppressed before the interpolation stage can expand its influence.

4.1.4 Stage 2: Bilinear Demosaicing

The task of Stage 2 is to reconstruct a full three-channel RGB image from the single-channel Bayer RAW data corrected by Stage 1. Each pixel position receives, through interpolation from same-color neighbors, the two color components it lacks. This step is necessary because standard pretrained perception models expect three-channel RGB input. Architecturally, Stage 2 receives corrected Bayer RAW from Stage 1 and delivers RGB to Stage 3 for final color adjustment. Its implementation is deliberately simple: a 3×3 bilinear interpolation window requiring only two lines of BRAM buffering, with interpolation arithmetic implemented using adders and shifters and no DSP multipliers. The pipeline is shown in Figure 4.6.

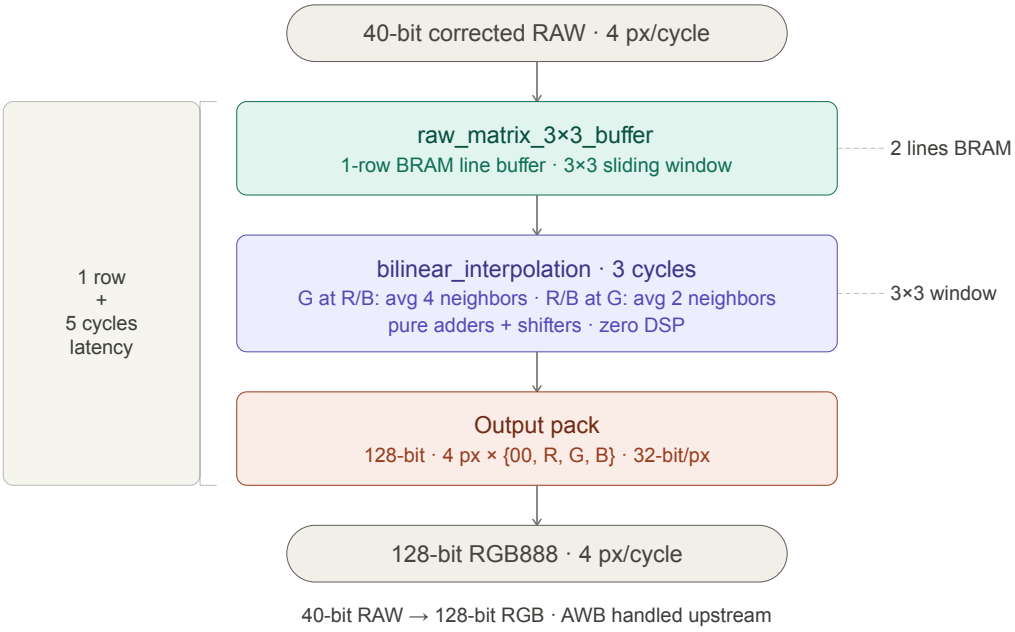

Figure 4.6: Demosaicing pipeline.

This stage therefore provides the required RGB format with minimal buffering and no multiplier usage.

4.1.5 Stage 3: Combined AWB and Gamma

Stage 3 is the final stage of the pixel data path and performs the two color adjustments needed to make the RGB output directly usable by downstream models. White balance correction compensates for the sensor's wavelength-dependent sensitivity using fixed gain ratios, while gamma correction reshapes the pixel value

distribution so that dark-region detail occupies a wider code-value range. In a conventional ISP, these two operations occupy separate pipeline stages. This design merges them into a single lookup by pre-combining the white balance gain into the gamma curve. Three 256-entry LUTs, one per color channel, store the merged AWB+gamma response, so that a single LUT access simultaneously corrects both color balance and tone distribution. The architecture is shown in Figure 4.7.

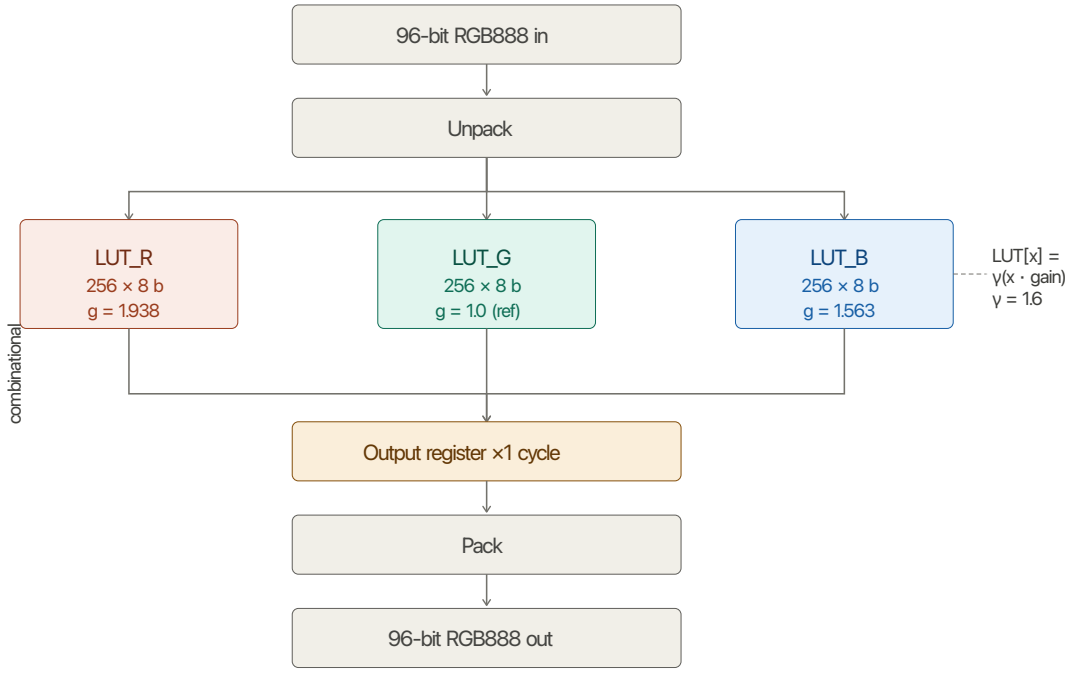

Figure 4.7: AWB and gamma correction pipeline.

The final pipeline stage fuses AWB gain application and gamma correction into a single lookup operation:

$$\mathrm{LUT}_c[x] = \Gamma\Big(\mathrm{clamp}(x \cdot g_c,\ 0,\ 255)\Big),$$

where Γ is the gamma transfer function and g_c is the calibrated channel gain. Each R, G, B channel has an independent LUT, requiring 768 bytes of distributed RAM. The 4-stage pipeline introduces 4 cycles of latency.

The gamma value is set to 1.6, lower than the standard sRGB value of 2.2. This produces weaker nonlinear compression, preserving more separation in dark and midtone regions. Since R, G, B channels independently undergo the same power-law transformation, color ratios are preserved and no additional color deviation is introduced.

4.2 Physical Setup

4.2.1 Camera Protective Enclosure

The camera module is a bare circuit board, vulnerable to vibration, dust, and accidental impact in vehicle-mounted testing. A protective enclosure (Figure 4.8) was designed and 3D-printed using PLA material with 2 mm wall thickness. The enclosure secures the circuit board via internal positioning slots, exposing only the lens and USB connector.

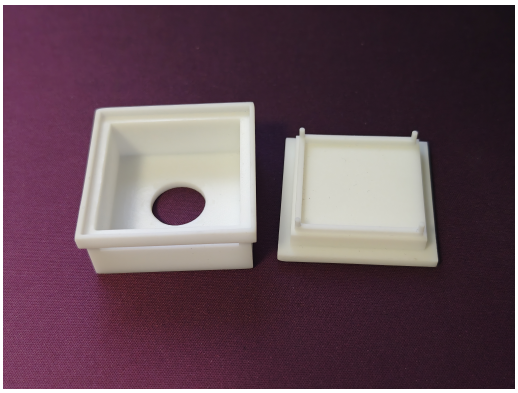

Figure 4.8: 3D-printed protective enclosure for the IMX291 camera module.

4.2.2 Vehicle Mounting

To capture the same scene simultaneously with the system camera and the reference camera, a suction-cup action-camera mount is used as the base, attached to the inside of the vehicle windshield. The DJI Action 5 Pro is fixed via its dedicated quick-release clamp. The IMX291 module is secured to the same mount using elastic bands, ensuring closely matched shooting angles and fields of view. The mounting setup is shown in Figure 4.9. The suction mount remained stable during the vehicle-mounted tests.

Figure 4.9: Camera mounting setup.

4.2.3 Development and Acquisition Environment

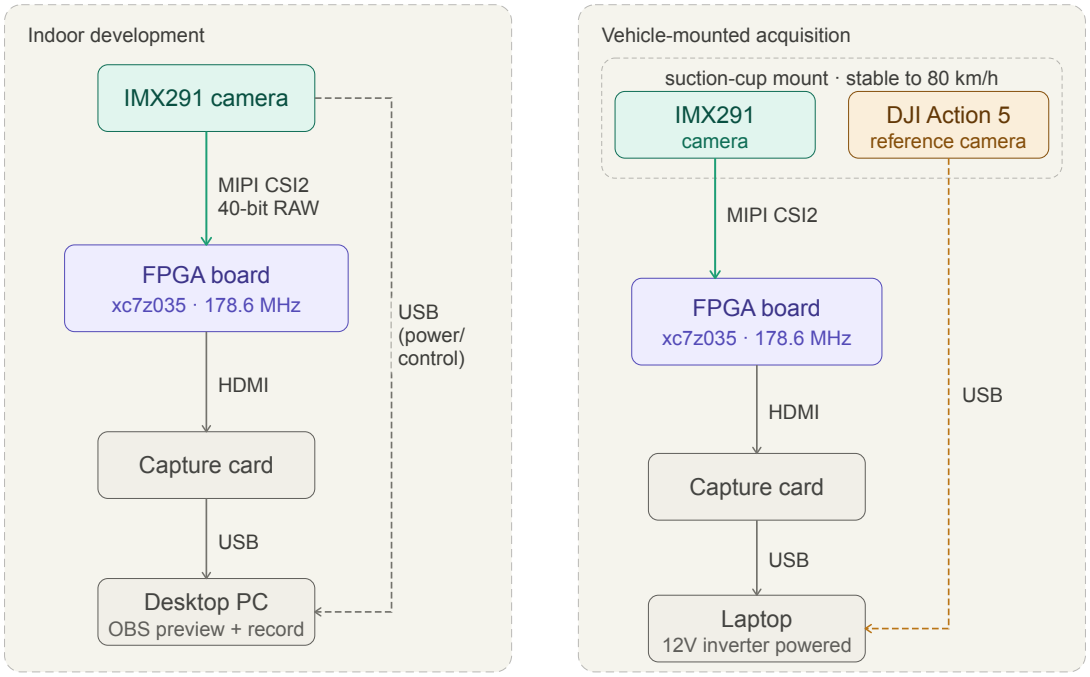

Figure 4.10: Development and acquisition environment.

Figure 4.10 illustrates the complete development and acquisition environment topology. During indoor development, the camera and FPGA development board connect to a desktop PC via USB and HDMI. The FPGA output passes through a video capture card for HDMI-to-USB conversion, with real-time preview and recording on the PC. For vehicle-mounted acquisition, the desktop PC is replaced by a laptop while maintaining the same connection topology: FPGA HDMI output to capture card to laptop USB input. The laptop is powered through a 12V automotive power inverter, enabling off-grid road acquisition. The actual indoor and in-vehicle setups are shown in Figure 4.11.

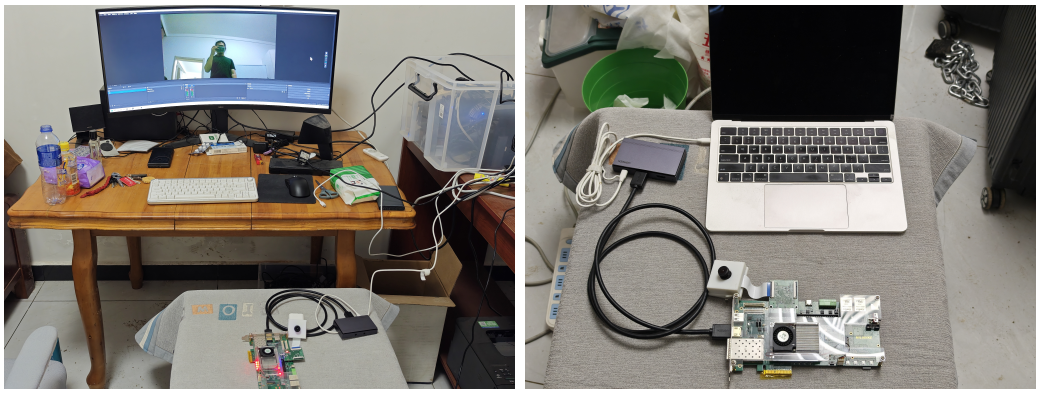

Figure 4.11: Development environment.

4.3 Downstream Validation Setup

Chapter 3 established the validation strategy: use style transfer to generate test images bearing the system's imaging style on existing annotated datasets, and compare downstream model accuracy against models trained on original data. This section specifies the experimental configuration.

4.3.1 CUT Style Transfer Setup

Two transfer directions are configured. The first direction transfers KITTI images to the style of Cityscapes-captured images, simulating the data distribution shift caused by different ISP configurations. The second direction transfers Cityscapes images to the style of a traditional HVS-oriented ISP pipeline, generating contrastive training data with noise and color shifts characteristic of less-optimized ISP processing. In both cases, the contrastive unpaired translation (CUT) model [35] is trained on unpaired images from the source and target domains. The output images retain the original pixel-level annotations, ensuring that any difference in model accuracy can be attributed solely to the change in imaging style.

4.3.2 Model Training Configuration

Two representative perception tasks are evaluated, as summarized in Table 4.5.

Table 4.5: Experimental configuration for downstream validation.

Task	Model	Dataset
Object Detection	YOLOv8m [36]	KITTI [38]
Semantic Segmentation	SegFormer-B0 [37]	Cityscapes [39]

For object detection, YOLOv8m is trained on real KITTI images and on CUT-transferred KITTI images under identical conditions. The detection task relies on high-level semantic features that are relatively robust to color shifts and mild noise, making it a suitable test of whether ISP style variations affect object-level recognition stability.

For semantic segmentation, SegFormer-B0 is trained on an aligned dataset of 1,000 Cityscapes images, including 800 training images and 200 validation images, with identical annotations. The original model trains on real Cityscapes images from 10 training cities; the style-transferred model trains on the same images after CUT transfer to an HVS-ISP style. Both use identical architecture, AdamW optimizer, training epochs, and geographically separated validation cities unseen during training. Semantic segmentation, as a pixel-wise dense prediction task, imposes higher requirements on local feature consistency, making it more sensitive to ISP-induced per-pixel style variations than object detection.

Every experiment follows a single-variable control principle: architecture, optimizer configuration, data split, and annotations are held identical; only the imaging style of the training data varies.

4.3.3 Functional Comparison

In parallel with the quantitative model evaluation, a qualitative functional comparison is conducted. The system output and a DJI Action 5 Pro reference camera record synchronized video of the same road scenes under both daytime and nighttime conditions. Frames are extracted and compared side by side. The comparison focuses on three dimensions: bloom suppression effectiveness around strong light sources, dark-region detail preservation, and overall color naturalness. The objective is not to claim superiority, but to verify that the system produces output of comparable visual adequacy to a professional consumer camera while achieving deterministic latency.

The corresponding quantitative and visual results are presented in Chapter 5.

5

Results

This chapter presents the experimental outcomes in three categories. Section 5.1 provides a visual comparison between the system output and a reference camera under normal and extreme lighting. Section 5.2 reports the CUT style transfer results. Sections 5.3 and 5.4 evaluate downstream model accuracy on KITTI object detection and Cityscapes semantic segmentation, respectively. Section 5.5 presents the FPGA synthesis results, including resource consumption, latency, and throughput. Section 5.6 summarizes the key findings.

5.1 Visual Comparison: System Output vs. Reference Camera

To establish a baseline, a comparison was first conducted under normal daylight conditions with moderate, uniform illumination. Figure 5.1 shows the same road scene captured by the DJI Action 5 Pro and the system output from the IMX291 sensor processed through the FPGA ISP pipeline, with synchronized frames extracted from video recordings. The figure consists of three parts arranged from left to right: the left panel shows the DJI Action 5 Pro full-frame reference, the center panel shows the full-frame system output, and the right panel presents a stacked zoomed comparison of the marked region, with the DJI crop above and the system crop below, for pixel-level detail inspection.

Figure 5.1: Visual comparison under normal daylight conditions. Left: DJI Action 5 Pro reference, center: system output, right: zoomed comparison of the marked region (DJI above, system output below).

Two differences are visually apparent. First, a subtle tone shift exists between the two images. The system output appears slightly brighter overall than the DJI reference, with more visible shadow detail. This difference originates from the two systems' different gamma values: our system uses $\gamma = 1.6$ to preserve more dark-region gray levels, whereas the DJI Action 5 Pro, as a consumer-grade camera, employs a value close to the sRGB standard of $\gamma \approx 2.2$, which applies more aggressive compression to dark regions. A lower gamma produces a gentler tonal curve, lifting midtones and shadows relative to a higher gamma. As a result, the system output renders dark areas, such as road surfaces and vehicles in shadow, brighter than the DJI reference, with correspondingly lower global contrast. Second, in the zoomed comparison on the right, the system output reveals mild luminance noise that is absent from the DJI reference, particularly in uniform regions such as the sky and road surface. This sensor noise is a direct consequence of omitting the denoising module from the ISP pipeline, a deliberate design decision grounded in the literature consensus that mild sensor noise does not harm, and may even act as a regularizer for, downstream neural networks [20].

Figures 5.2 and 5.3 extend the comparison to more challenging lighting conditions. Frames were extracted from synchronized video recordings of the same scenes.

Figure 5.2: Daytime scene comparison.

Figure 5.3: Nighttime scene comparison.

Under daytime conditions, the system output achieves comparable overall image quality to the reference camera, with natural color reproduction and adequate dynamic range. Under nighttime conditions, bloom suppression effectively reduces the halo area around strong light sources, and the constraint-based AEC preserves dark-region texture detail that would otherwise be lost through conventional global gain reduction.

5.2 CUT Style Transfer Results

Two CUT-based style transfer tasks were performed to generate training data with different ISP imaging styles. Figures 5.4 and 5.5 show representative examples of the style transfer results.

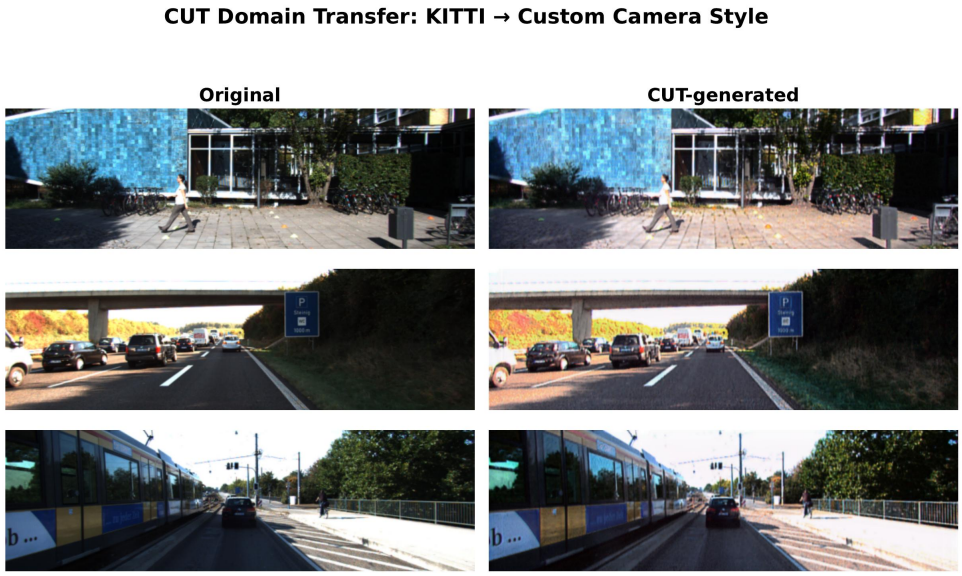

Figure 5.4: KITTI dataset images before and after style transfer.

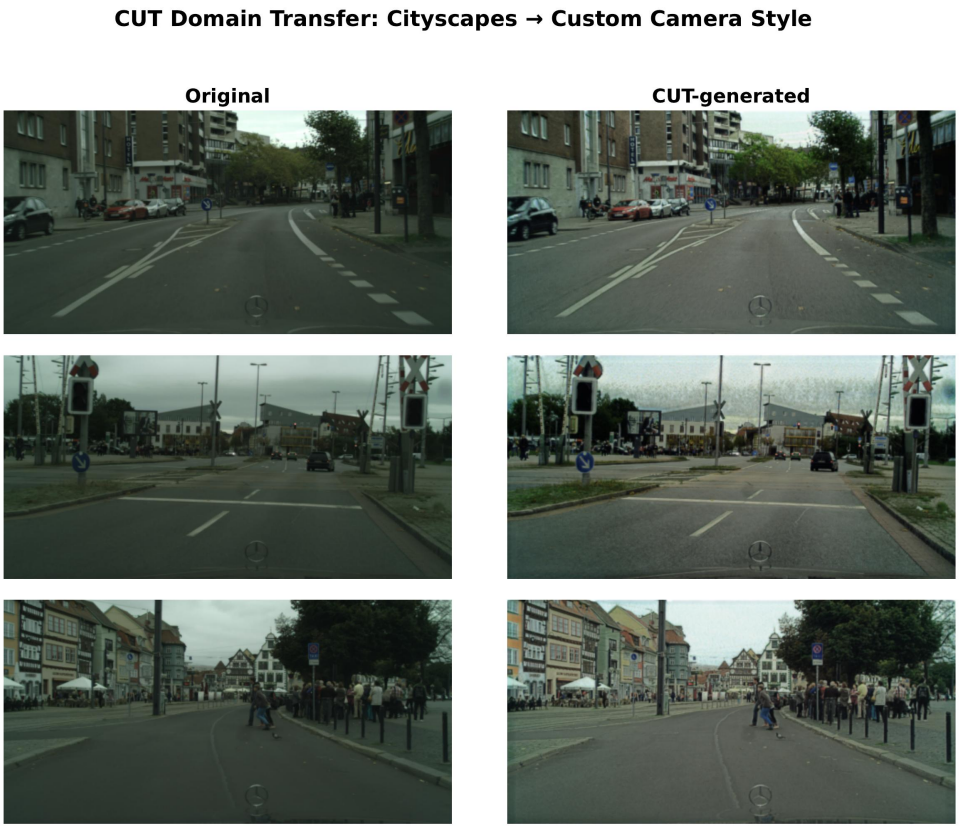

Figure 5.5: Cityscapes dataset images before and after style transfer.

The style-transferred images preserve the scene semantic content and ground-truth annotations of the original images. Changes are confined to imaging style attributes: color response, noise distribution, and tone mapping curves. The kitti_cut transfer introduces noticeable texture differences reflecting the distinct ISP configurations of the two datasets, while the city_cut transfer adds noise and color shifts characteristic of traditional HVS-oriented ISP pipelines.

5.3 KITTI Object Detection Experiment

Two YOLOv8m detection models were trained under identical conditions except for the training image style. The original model was trained on real KITTI images. The fake model was trained on the same images after kitti_cut style transfer (KITTI to Cityscapes style). Both models used identical architecture, optimizer configuration, validation split, and ground-truth annotations.

5.3.1 Training Loss Comparison

Figure 5.6 presents the training metrics of the YOLOv8m detection model on the original KITTI dataset and its CUT style-transferred counterpart. Box loss measures the error between predicted and ground-truth bounding box positions; a lower value indicates more accurate localization. Cls loss measures the classification error;

a lower value indicates more accurate category prediction. DFL loss quantifies the distribution focal loss for bounding box regression; a lower value indicates more stable box regression. mAP@50 and mAP@50-95 represent mean average precision at an IoU threshold of 0.50 and averaged over IoU thresholds from 0.50 to 0.95, respectively. Higher values indicate better overall detection performance, with mAP@50-95 being the more stringent metric. Precision measures what fraction of predicted detections correspond to real objects, and recall measures what fraction of real objects are successfully detected. As shown, all loss metrics for both datasets decrease steadily throughout training, while mAP, precision, and recall rise consistently, indicating normal model convergence in both conditions. The CUT-transferred data yields slightly lower metrics than the original data on some indicators, but the overall trends are consistent. This suggests that the object position, category, and contour information are well preserved after simulating the imaging style of the proposed ISP pipeline, and no significant negative impact is introduced for downstream detection training.

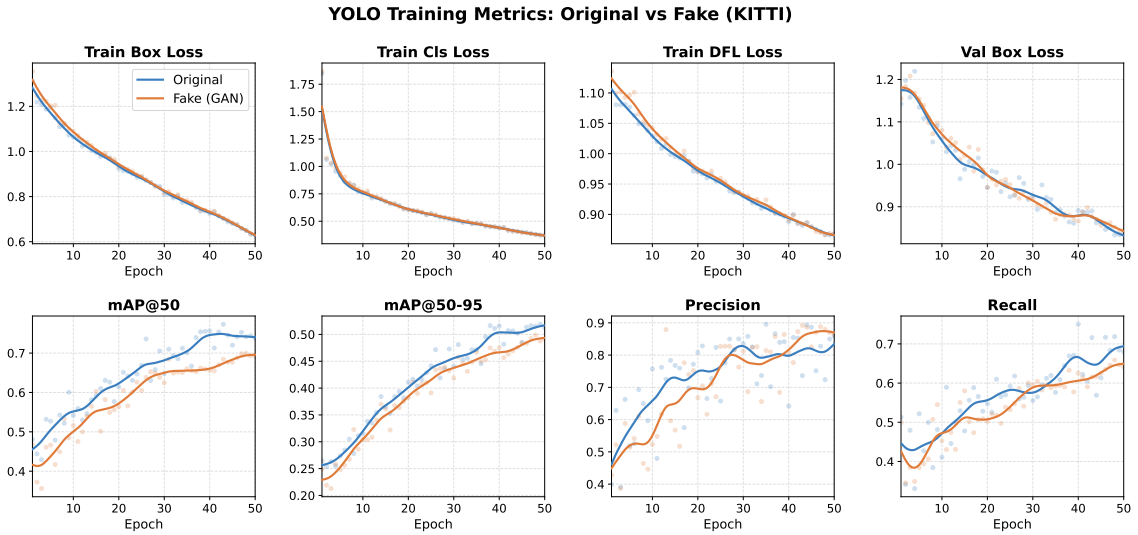

Figure 5.6: YOLOv8m training loss curves for original and style-transferred KITTI images.

5.3.2 Detection Visualization

Figure 5.7 presents side-by-side detection comparisons on identical test images.

Figure 5.7: Detection results: original-trained model (left) versus style-transfer-trained model (right).

Detection bounding boxes, class labels, and confidence scores are consistent between the two models. No systematic shift in detection quality is observed. The results indicate that the ISP style variations introduced by CUT transfer do not harm object detection performance. High-level semantic features are generally robust to the color and texture changes introduced by different ISP configurations.

5.4 Cityscapes Semantic Segmentation Experiment

Two SegFormer-B0 segmentation models were trained using an aligned dataset of 1,000 images (800 train, 200 validation) with identical annotations. The original model was trained on real Cityscapes images from 10 training cities. The fake model was trained on city_cut style-transferred images (Cityscapes to traditional HVS-ISP style). Both models used identical architecture, optimizer (AdamW), and validation cities (8 geographically separated cities unseen during training).

5.4.1 Training Loss Analysis

Figure 5.8 presents the training loss curves of the SegFormer-B0 segmentation model on the original Cityscapes dataset and its CUT style-transferred counterpart. Loss represents the error between the model's predictions and the ground-truth labels; a lower value indicates better fitting of the model to the segmentation targets. As shown, the loss for both datasets decreases progressively with the number of training epochs, confirming that the model is able to learn and converge normally on both the

original images and those simulating the imaging style of the proposed ISP pipeline. The CUT-transferred data exhibits a slightly higher loss in the early training stage, indicating that the style variation introduces some additional learning difficulty, but the curves of the two conditions gradually converge in later epochs. This suggests that CUT primarily alters the visual appearance of the images without substantially disrupting the semantic information required for segmentation.

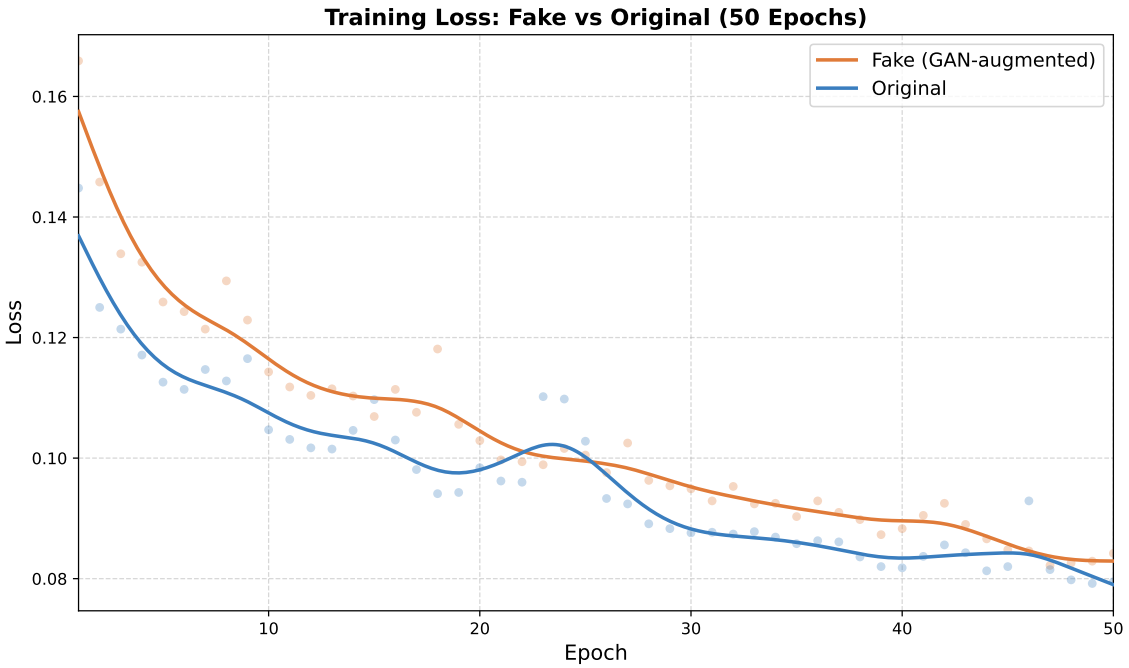

Figure 5.8: SegFormer-B0 training loss curves for original and style-transferred Cityscapes images.

Table 5.1: SegFormer-B0 training loss comparison at selected epochs.

Epoch	Fake Training Loss	Original Training Loss	Difference
1	0.166	0.145	+14.5%
10	0.114	0.105	+8.6%
30	0.095	0.088	+8.0%
50	0.084	0.080	+5.0%

Three phases are observed in the training dynamics. In the initial convergence phase (Epochs 1–10), the fake model starts with approximately 14% higher loss, reflecting the distribution shift introduced by style transfer. In the catch-up phase (Epochs 10–30), the loss gap narrows from 14% to 8%, showing the model's capacity to gradually adapt to the transferred data distribution. In the final convergence phase (Epochs 30–50), the gap narrows further to approximately 5%, and both models reach comparable convergence accuracy. No divergence or convergence failure is observed in the fake training condition.

5.4.2 Segmentation Visualization

Figures 5.9 and 5.10 show segmentation results from both models on identical test images.

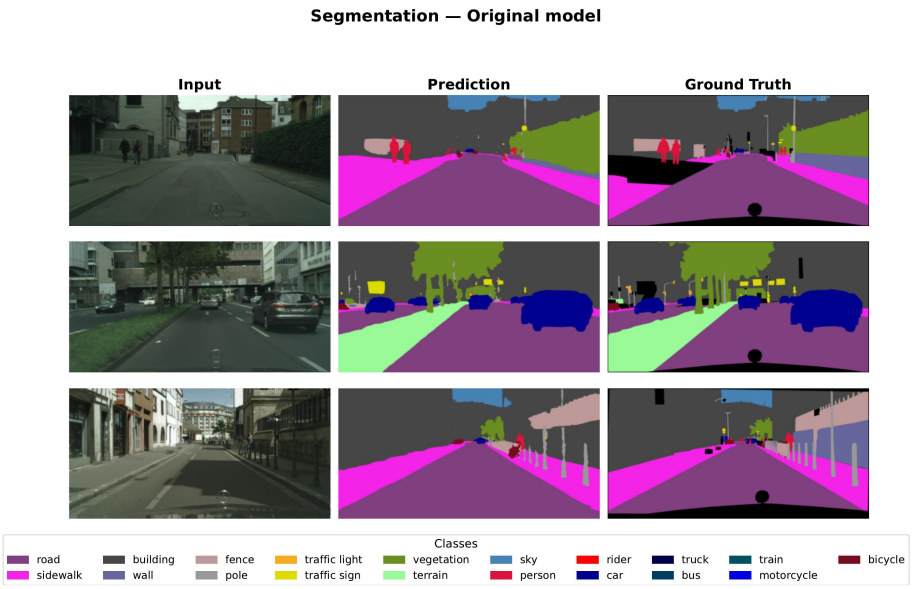

Figure 5.9: SegFormer-B0 segmentation results: model trained on original Cityscapes images.

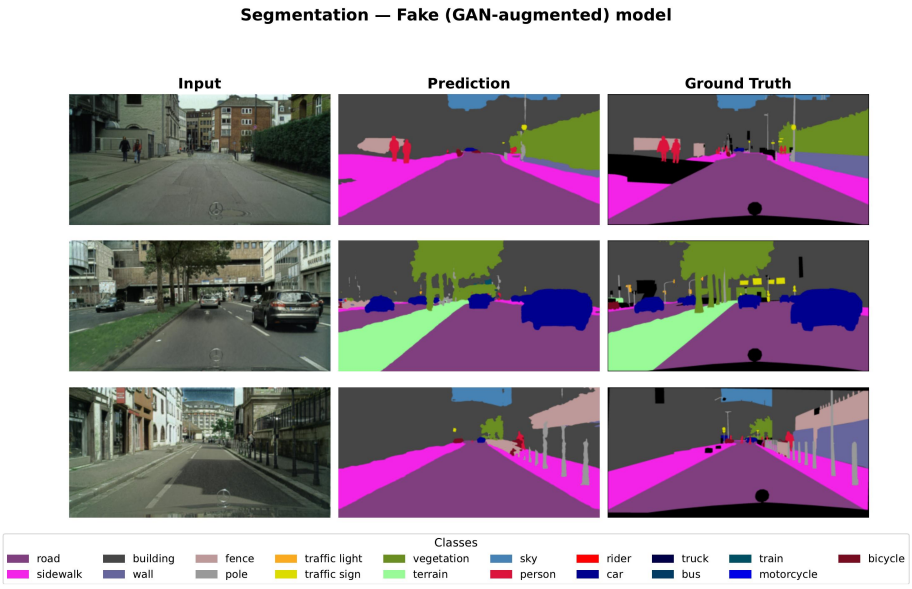

Figure 5.10: SegFormer-B0 segmentation results: model trained on city_cut style-transferred images.

No systematic differences are observed between the two models in segmentation mask quality. Class boundaries, small-object recognition (distant pedestrians, traffic signs), and overall segmentation coherence are comparable. Semantic segmen-

tation, as a pixel-wise dense prediction task, shows good adaptability to the data distribution shift introduced by CUT transfer.

5.5 FPGA Implementation Results

5.5.1 Core Logic Resources

The ISP data-path core modules (bloom affine, demosaicing, combined AWB and gamma, and the AEC side channel) were synthesized standalone on a Xilinx xc7z035ffg676-2 device. The synthesis achieved a clock frequency of 178.6 MHz.

Before presenting the numerical results, it is necessary to distinguish two concepts of latency that are often conflated in pipeline performance discussions. **Frame processing time** is the interval from the first pixel of a frame entering the pipeline to the last pixel of that frame exiting; it reflects the system's sustained throughput and determines the theoretical maximum frame rate. **Group delay** (first-pixel latency) is the interval from the first pixel entering to the first corresponding output pixel emerging; it reflects the pipeline's end-to-end response speed and is determined entirely by line-buffer fill depth and pipeline register stages, independent of frame height and scene content. Both quantities scale linearly with the clock period T_{clk}, so a higher clock frequency proportionally improves both. The distinction matters for autonomous driving: frame processing time governs whether the system can keep up with the incoming video stream, while group delay governs how quickly a sudden scene change, such as an obstacle appearing or a headlight flaring, is reflected in the output.

The pipeline processes 4 pixels per clock cycle. For a 1080p frame (1920×1080 pixels), the number of clock cycles required to stream one complete frame through the pipeline is

$$N_{\text{frame}} = \frac{1920}{4} \times 1080 = 480 \times 1080 = 518{,}400 \text{ cycles.}$$

At the achieved clock frequency, the time required to process one full frame is therefore

$$t_{\text{frame}} = \frac{N_{\text{frame}}}{f_{\text{clk}}} = \frac{518{,}400}{178.6 \times 10^6} \text{ s} \approx 2.903 \text{ ms.}$$

The corresponding maximum frame rate, assuming negligible vertical blanking overhead, is

$$\text{FPS}_{\text{max}} = \frac{1}{t_{\text{frame}}} \approx 344 \text{ fps,}$$

which exceeds the 30/60 fps requirements of typical video applications by more than an order of magnitude.

The per-pixel latency, meaning the delay between a pixel entering the pipeline and its corrected RGB value emerging at the output, is the sum of line-buffer fill times and pipeline register depths across the three stages. Stage 1 requires 5 rows of buffering to fill the 11×11 box filter window, plus 5 register stages in its computational pipeline. Stage 2 adds 1 row for the 3×3 demosaicing window and 5 pipeline stages. Stage 3

contributes 4 LUT pipeline stages and 1 format-bridge register. The total is 6 lines and 15 cycles. Expressed as an absolute time at 178.6 MHz,

$$t_{\text{pixel}} = (6 \times 480 + 15) \times \frac{1}{178.6 \times 10^6}\ \text{s} \approx 16.2\ \mu\text{s}.$$

This latency is deterministic: it depends only on the image width and clock frequency, not on scene content, and remains fixed from frame to frame. The latency and throughput metrics are summarized in Table 5.2, and the resource consumption is summarized in Table 5.3.

Table 5.2: Core ISP logic latency and throughput.

Metric	Value
Data-path latency	6 lines + 15 cycles
First-pixel latency	∼6.02 lines
Clock frequency	178.6 MHz
1080p frame processing time	2.903 ms
1080p maximum frames per second (FPS)	∼344

Table 5.3: Core ISP logic resource consumption.

Resource	Used (utilization)
Slice LUTs	2,768 (1.61%)
Slice Registers	2,596 (0.76%)
Block RAM Tiles	45 (9.00%)
DSP48E1 Slices	13 (1.44%)

BRAM consumption is dominated by line buffering: 17 tiles for the bloom box filter, 3 tiles for demosaicing, and 4 tiles for the AEC histogram, totaling 45 tiles. DSP consumption arises from the box-filter division approximation (4 slices), the excess gain model multiplication (4 slices), gain composition multiplication (4 slices), and the gamma LUT (1 slice).

5.5.2 Full System Resources

Including all system infrastructure (MIPI CSI-2 receive with PHY, frame DMA with double buffering, video timing controller, AXI4-Stream video output bridge, HDMI transmission, ARM processing system, and clock/reset management), the full system synthesis results are shown in Table 5.4.

Table 5.4: Full system synthesis results including all I/O and bus infrastructure.

Resource	Used	Available	Utilization
Slice LUTs	6,354	171,900	3.70%
Slice Registers	7,398	343,800	2.15%
Block RAM Tile	60	500	12.0%
DSP48E1	13	900	1.44%

All resource utilization rates remain below 13%, and the 344 fps throughput far exceeds the 30/60 fps requirements of typical video applications. Over 96% of LUTs, 88% of BRAM, and 98% of DSP slices remain available for future extensions.

5.6 Summary of Findings

The experimental results collectively support the following conclusions. First, visual comparison with a professional action camera indicates that the system produces output of comparable quality under both daylight and nighttime conditions with strong light sources, while achieving deterministic latency that consumer cameras cannot provide. Second, the style variations and noise characteristics introduced by the system's ISP processing do not harm downstream model training or inference accuracy. Detection models (YOLOv8m) trained on style-transferred data match the accuracy of models trained on original data, and segmentation models (SegFormer-B0) converge to comparable final accuracy despite initially slower convergence. Third, the ISP pipeline achieves efficient FPGA implementation with resource utilization below 13% across all categories while delivering 344 fps throughput, far exceeding real-time requirements.

6

Discussion

Building on the design and results presented in Chapters 4 and 5, this chapter discusses the main design decisions, validation rationale, and limitations of the proposed system. Section 6.1 discusses the advantage of Bayer-domain preprocessing. Section 6.2 examines the rationale and boundaries of the style-transfer validation strategy. Section 6.3 discusses the simplicity-first engineering philosophy through two design examples. Section 6.4 examines implementation trade-offs. Section 6.5 identifies limitations and future work directions. Section 6.6 addresses ethical considerations.

6.1 Bayer-Domain Preprocessing Strategy

Two of the most critical operations in this design, AEC global gain and bloom halo suppression, are applied in the Bayer RAW domain rather than in the traditional RGB domain. This is not merely a pipeline-ordering adjustment. It directly addresses the diffusion of saturated energy through demosaicing interpolation.

The local gain computation used for bloom suppression compares neighborhood brightness against the pixel's own value. This idea is related to earlier work on local lightness and tone mapping. Land and McCann [7] showed in Retinex theory that perceived brightness depends on local relationships rather than absolute luminance alone. Reinhard et al. [34] later used local neighborhood comparisons in photographic tone reproduction. Those works were designed for human perception. In this thesis, the same general principle is repurposed for machine-vision-oriented preprocessing: the objective is to suppress halo energy before it can diffuse through demosaicing and contaminate local contrast cues used by downstream models.

In a Bayer CFA sensor, the state of a bloom region is not necessarily uniform across color channels. Green pixels are sampled more densely and may saturate first, while red and blue pixels may still contain useful information. In a traditional pipeline, demosaicing is completed before many RGB-domain corrections are applied. By that point, saturated values have already been interpolated into neighboring color positions, increasing the effective bloom footprint. Later gain reduction cannot undo this diffusion.

Applying gain suppression in the Bayer domain addresses the problem before interpolation occurs. Linear gain scaling is compatible with interpolation, so applying gain before or after demosaicing can produce equivalent results when values remain

unsaturated. The critical advantage is therefore temporal: Bayer-domain processing acts before saturated energy is spread by demosaicing. This is the main reason Stage 1 in Chapter 4 combines BLC, AEC gain, and local bloom suppression before the demosaicing stage.

A natural question is whether a more sophisticated AEC metering strategy could achieve the same effect without changing the pipeline placement. As discussed in Chapter 2, consumer and industrial ISPs can use center-weighted, multi-zone, or spot metering [32]. These strategies are useful for photographic exposure control, but they do not directly distinguish optical artifacts from scene content [41]. A saturated bloom region still elevates the brightness statistics of the zones it overlaps, and metering cannot recover pixel values already lost to saturation or prevent later demosaicing diffusion. Bayer-domain suppression therefore provides a structural advantage: the AEC operates on cleaner statistics because bloom is reduced before it strongly affects later processing.

6.2 Style-Transfer Validation

6.2.1 Rationale for the Validation Strategy

Chapter 3 introduced style transfer as the downstream validation strategy. The motivation is single-variable control. Ideally, two datasets would differ only in ISP style while sharing the same scene content and annotations. In practice, this is difficult to achieve through direct dual-camera capture. Two cameras cannot occupy exactly the same physical position, and differences in lens, field of view, distortion, sensor response, and timing would all become confounding variables. If the two datasets also require separate annotation, annotation differences become entangled with ISP differences.

A self-collected and manually annotated dataset would also be difficult to justify. High-quality object and pixel-level annotation is expensive, and a small self-collected dataset would not match the diversity and consistency of established benchmarks. For this reason, the thesis uses contrastive unpaired translation (CUT) to transfer imaging style onto existing annotated benchmark images. The original and style-transferred images share the same labels, so annotation quality is held constant while imaging style is varied.

The validation logic should be interpreted conservatively. CUT is a lossy mapping: it can modify color response, texture statistics, and tone distribution, but it cannot add new valid scene information. Therefore, if models trained on style-transferred data achieve accuracy comparable to models trained on original data, the result supports the conclusion that the simulated ISP-style variation does not substantially harm downstream model training. This does not prove that CUT is a perfect physical camera model; rather, it provides a controlled and repeatable approximation for testing sensitivity to imaging-style changes.

6.2.2 Validity Boundaries of CUT-Based Validation

The main limitation of CUT-based validation is fidelity. CUT-transferred images are generated by a learning model, not by the physical optics and electronics of the actual camera. The transferred images can approximate differences in color response, noise appearance, and tone mapping, but they cannot exactly reproduce sensor noise, lens flare, quantization behavior, or temporal effects of real video capture.

For this reason, the results should not be read as a complete replacement for real data collection. They establish that the downstream models remain robust under a controlled style shift similar to the one introduced by the proposed ISP pipeline. They do not fully quantify performance under all real-world lighting, motion, sensor-noise, and lens-artifact conditions. Real synchronized capture and direct hardware-in-the-loop evaluation remain necessary future validation steps.

This boundary does not invalidate the method. The key value of CUT-based validation is that it isolates imaging style while preserving annotations. It is therefore useful for testing whether ISP-induced appearance changes are likely to harm model accuracy, but its conclusions should be combined with the visual and hardware results in Chapter 5, rather than interpreted alone.

6.3 Design Philosophy: Simple versus Complex

The final architecture is intentionally simple: a three-stage streaming pipeline with a side-channel AEC, bilinear demosaicing, and AWB plus gamma correction through small LUTs. The core logic consumes only 2,768 LUTs, less than 2% of the target device resources. This result was not assumed at the start; it emerged from comparing simple modules with more complex alternatives.

6.3.1 Demosaicing: Cost of Complex Alternatives

Demosaicing provides the clearest example. Bilinear interpolation requires only a 3×3 window, two lines of BRAM buffering, adders, and shift operations. It uses no DSP multipliers. More sophisticated edge-adaptive methods can improve conventional image-quality metrics by preserving edges and reducing false color, but they require more line buffering, additional gradient logic, and substantially more LUTs and DSP resources.

For this thesis, the important question is not whether a complex demosaicing algorithm improves visual quality in isolation, but whether it improves the downstream tasks enough to justify its hardware cost. Based on the module-effectiveness analysis in Chapter 2, demosaicing is necessary, but the additional benefit of complex demosaicing for downstream detection and segmentation is limited compared with its resource cost. The bilinear solution therefore matches the simplicity-first design philosophy: it provides the required RGB output with deterministic timing and low resource consumption.

6.3.2 AEC Control: Robustness over Complexity

AEC gain control provides a second example. During early development, a PID-based controller was considered as an alternative to the constraint-based decision policy. PID control offers a familiar closed-loop framework, but it is sensitive to scene-dependent brightness statistics. Under bloom, saturated pixels can inflate the measured brightness and drive the integral term in an undesirable direction. When the bloom source then leaves the frame, the accumulated correction can produce visible gain swings.

In practice, this makes PID tuning scene-dependent. Parameters that behave acceptably in daytime driving may oscillate under nighttime urban conditions with headlights and street lamps. The final constraint-based policy avoids this problem. It uses percentile-based thresholds, hard gain bounds, a five-state FSM, an IIR low-pass filter, and a slew-rate limiter. This approach is less mathematically general than PID control, but it is more predictable for the tested scenes and easier to implement in a streaming FPGA pipeline. The design choice therefore reflects the same principle: a simpler method is preferred when it satisfies the specification with lower implementation risk.

6.4 Implementation Trade-offs

6.4.1 Clock Frequency

The synthesized clock frequency of 178.6 MHz enables a theoretical throughput of 344 fps at 1080p, which is far above the 30 fps target. Reducing the clock frequency might appear attractive, but three considerations argue against doing so in this implementation.

First, the resource consumption is already low. The core logic occupies 1.6% of LUTs, 0.8% of registers, 9.0% of BRAM, and 1.4% of DSP slices. There is no strong resource pressure that would justify relaxing timing constraints.

Second, the 344 fps value assumes an ideal streaming model. In practice, the MIPI CSI-2 interface includes blanking intervals, sensor timing variation, and clock-domain crossing effects. Additional frequency headroom helps absorb such irregularities and reduces the risk of backpressure or frame drops.

Third, validating a lower-frequency implementation would require re-running timing closure and physical camera tests. Given the already low resource usage and stable operation of the current implementation, the engineering effort would provide limited practical benefit.

More complex designs would also increase timing pressure. Deeper pipelines, more conditional control, and wider arithmetic paths can reduce the maximum achievable clock frequency or require additional registers. For a low-latency video system, preserving timing margin is therefore a practical design goal, not only a performance preference.

6.4.2 Streaming versus Block-Parallel Processing

The proposed architecture uses streaming processing because it matches the data source. A MIPI CSI-2 camera transmits pixels line by line, so pixels naturally arrive in sequential order. A streaming FPGA pipeline can process them as they arrive, requiring only the line buffers needed by local windows and no full-frame buffer.

Block-parallel processing could provide broader simultaneous access to image regions, but it would require larger buffers or external memory. This introduces DDR access, bus arbitration, clock-domain synchronization, and less predictable latency. Such complexity is difficult to justify when the required operations, local bloom filtering, demosaicing, and histogram accumulation, can all be implemented in-stream.

The design therefore combines streaming data flow with lightweight side-channel statistics. The bloom box filter provides local neighborhood information through line buffering, while the AEC side channel accumulates frame-level histograms and applies the resulting gain to the next frame. This creates at most a one-frame control delay while preserving fixed data-path latency. The approach is a deliberate trade-off: it sacrifices some global immediacy in exchange for low resource use, deterministic timing, and architectural simplicity.

6.5 Limitations and Future Work

This work has several limitations.

First, parameter calibration is sensor-dependent. The fixed gamma value ($\gamma = 1.6$) and fixed AWB gains (R=1.28×, B=1.48×) were calibrated for the Sony IMX291 sensor and its lens module. Replacing the sensor or lens would change the spectral response and optical characteristics, requiring recalibration. This limitation is a consequence of the low-cost fixed-parameter design.

Second, the CUT-based validation has a fidelity boundary. As discussed in Section 6.2, style-transferred images are not physical sensor captures. They approximate imaging-style changes but do not exactly reproduce real sensor noise, lens artifacts, or temporal video effects. The validation therefore supports downstream robustness to style variation, but it does not replace direct evaluation with real captured datasets.

Third, bloom suppression has a spatial-range limit. The current implementation uses an 11×11 box filter, corresponding to an effective radius of approximately 5 pixels. Larger halos, such as those produced by direct sunlight or wide-angle lens flare, may not be fully suppressed. Increasing the window size would extend the suppression range but would also increase line-buffer usage.

Fourth, validation is limited to a single camera stream. Production autonomous-driving systems often rely on multiple cameras for surround-view perception. Inter-camera exposure consistency, frame synchronization, and cross-camera gain control are outside the scope of the current design.

Fifth, end-to-end latency measurement is incomplete. The per-pixel latency of 16.2 μs and the frame processing time of 2.903 ms are theoretical values derived

from the clock frequency and pipeline depth. Direct hardware measurement using an oscilloscope, photodiode, or high-speed camera would be needed to quantify the complete sensor-to-HDMI latency and any jitter introduced by interface handshaking, DDR access, or AXI bus contention.

Future work can proceed in three directions. First, multi-camera synchronized surround-view processing would extend the current single-stream design to practical autonomous-driving camera configurations. Second, multi-exposure HDR fusion could further extend dynamic range in severe backlit scenes where one exposure cannot preserve both highlights and shadow details. The current throughput headroom makes this a practical extension, but it would require exposure sequencing and lightweight frame fusion. Third, direct hardware latency measurement would provide measured data for evaluating safety-relevant timing behavior.

6.6 Ethical Considerations

6.6.1 Generative AI Usage

Generative AI tools were used in selected parts of this work. The tools employed were Claude Code, powered by Claude 4.6 Sonnet, and DeepSeek V4. For technical development, AI assisted in generating selected functionally simple Verilog RTL modules and in locating potential issues during debugging. For literature research, AI was used for literature search and preliminary screening. For report writing, AI assisted with grammar polishing, paragraph restructuring, and formatting improvements. Several block diagrams in this thesis were generated by AI based on the authors' design specifications, under human guidance.

All AI-generated or AI-modified content was reviewed and revised by the authors. The authors bear full responsibility for all technical claims, design decisions, experimental results, and academic content in this thesis.

6.6.2 Data Privacy and Safety

Image preprocessing for autonomous driving must not introduce misleading modifications to safety-relevant visual features, such as the apparent position, shape, or category of objects used by downstream perception models. The proposed pixel data path is deterministic and non-generative: it uses gain adjustment, local filtering, demosaicing, and LUT-based tone mapping. It does not synthesize semantic content or perform object-level editing. This reduces the risk of creating objects that were not present in the scene or removing safety-relevant structures from the image.

The video data collected for this work consists of road traffic scenes captured from a forward-facing camera mounted inside a vehicle. The footage records road environments, vehicles, traffic infrastructure, and lighting conditions at normal driving distances. The system was not designed for identifying people, recognizing faces, reading license plates, or tracking individuals. The CUT style-transfer validation

pipeline modifies imaging characteristics such as color distribution, texture, and luminance response, and does not perform identity inference.

Data collection and handling were designed to respect privacy principles under the General Data Protection Regulation (GDPR) and relevant Swedish guidance on camera surveillance in public spaces [42], [43]. The collected footage was used only for technical validation of the image processing pipeline and downstream perception robustness.

7

Conclusion

This chapter summarizes the principal results, revisits the original contributions, and outlines directions for future work.

This thesis presented the design, implementation, and evaluation of an FPGA-based visual degradation processing system for automotive camera perception under extreme lighting conditions. The system integrates a constraint-based AEC loop, Bayer-domain bloom suppression, and a pruned three-stage ISP pipeline optimized for machine vision rather than human viewing.

Experimental validation through CUT-based style transfer demonstrated that the imaging style variations and noise characteristics introduced by the system do not harm model training or inference accuracy. Object detection models trained on style-transferred data matched the accuracy of those trained on original data. Semantic segmentation models achieved comparable final accuracy, with the loss gap narrowing from 14.5% to 5.0% over 50 training epochs. Qualitative side-by-side comparison with a professional action camera confirmed comparable image quality under both daytime and nighttime conditions. At the same time, the system achieves substantially lower and deterministic pipeline latency, supporting the engineering objective of producing visually adequate output for downstream perception with hard real-time guarantees.

The system's low and deterministic latency, high throughput, minimal resource utilization, and validated compatibility with downstream perception models demonstrate that task-specialized ISP pipelines can simultaneously achieve efficient hardware implementation and maintained machine vision performance.

Future work includes multi-camera synchronization for surround-view processing, integration of multi-exposure HDR fusion, and real-time in-vehicle deployment testing.

Table 7.1 summarizes the principal quantitative outcomes.

Table 7.1: Summary of key results.

Latency and Throughput	
Per-pixel processing latency	6 lines + 15 cycles
1080p frame processing time	2.903 ms
Maximum 1080p throughput	344 fps
FPGA Resource Consumption	
Core logic LUTs	2,768 (1.6%)
Core logic BRAM	45 tiles (9.0%)
Core logic DSP	13 slices (1.4%)
Full system LUTs	6,354 (3.7%)
Full system BRAM	60 tiles (12.0%)
Full system DSP	13 slices (1.4%)
Downstream Model Validation	
Object detection	YOLOv8m on KITTI remained functional, with loss curves nearly identical to the original-data baseline.
Semantic segmentation	SegFormer-B0 on Cityscapes remained functional, with the loss gap reduced from 14.5% to 5.0% over 50 epochs.

Bibliography

[1] Slate, "How Tesla's software update fixed a deadly flaw in Autopilot," https://slate.com/technology/2016/09/how-teslas-software-update-fixed-a-deadly-flaw-in-autopilot.html, 2016, accessed: Jun. 13, 2026.

[2] D. Hull and C. Trudell, "A fatal Tesla crash shows the limits of full self-driving," https://www.bloomberg.com/features/2025-tesla-full-self-driving-crash/, 2025, bloomberg News, Jun. 4, 2025; corrected Jun. 12, 2025; accessed: Jun. 13, 2026.

[3] Tesla Motors Club, "FSD runs red light when have sun glare," https://teslamotorsclub.com/tmc/threads/fsd-runs-red-light-when-have-sun-glare.316413/, 2023, forum thread; accessed: Jun. 13, 2026.

[4] United Nations Economic Commission for Europe, "UN regulation no. 46: Uniform provisions concerning the approval of devices for indirect vision and of motor vehicles with regard to the installation of these devices," 2015, world Forum for Harmonization of Vehicle Regulations (WP.29).

[5] *Motor vehicles – Devices for indirect vision*, Standardization Administration of China Std. GB 15 084–2022, 2022.

[6] K. Zuiderveld, "Contrast limited adaptive histogram equalization," in *Graphics Gems IV*, P. S. Heckbert, Ed. San Diego, CA: Academic Press Professional, 1994, pp. 474–485.

[7] E. H. Land and J. J. McCann, "Lightness and Retinex theory," *Journal of the Optical Society of America*, vol. 61, no. 1, pp. 1–11, 1971.

[8] X. Guo, Y. Li, and H. Ling, "LIME: Low-light image enhancement via illumination map estimation," *IEEE Transactions on Image Processing*, vol. 26, no. 2, pp. 982–993, 2017.

[9] C. Wei, W. Wang, W. Yang, and J. Liu, "Deep Retinex decomposition for low-light enhancement," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2018, oral.

[10] C. Guo, C. Li, J. Guo, C. C. Loy, J. Hou, S. Kwong, and R. Cong, "Zero-reference deep curve estimation for low-light image enhancement," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 1780–1789.

[11] Y. Cai, H. Bian, J. Lin, H. Wang, R. Timofte, and Y. Zhang, "Retinexformer: One-stage Retinex-based transformer for low-light image enhancement," in

Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2023, pp. 12 504–12 513.

[12] C. Li, C. Guo, L. Han, J. Jiang, M.-M. Cheng, J. Gu, and C. C. Loy, "Low-light image and video enhancement using deep learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 12, pp. 9396–9416, 2022.

[13] J.-W. Park, H. Lee, B. Kim, D.-G. Kang, S.-O. Jin, and H.-J. Kim, "A low-cost and high-throughput FPGA implementation of the Retinex algorithm for real-time video enhancement," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 1, pp. 101–114, 2020.

[14] B. B. Upadhyay and K. Sarawadekar, "A low-cost FPGA implementation of Retinex-based low-light image enhancement algorithm," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2024.

[15] T. Kryjak, K. Blachut, H. Szolc, and M. Wasala, "Real-time CLAHE algorithm implementation in SoC FPGA device for 4k UHD video stream," *Electronics*, vol. 11, no. 14, p. 2248, 2022.

[16] C. Wang and J. Xu, "Lightweight CNN-based low-light-image enhancement system on FPGA platform," *Neural Processing Letters*, vol. 55, pp. 8023–8039, 2023.

[17] F. M. Siddiqui, M. Russell, B. Bardak, R. Woods, and K. Rafferty, "IPPro: FPGA-based image processing processor," in *Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS)*, 2014, pp. 1–6, iEEE Xplore Document No. 6861620.

[18] T. Bilal, B. Amjad, T. Iqbal, B. Zafar, K. Usman, and S. I. Bhatti, "InfiniteISP: An open-source hardware image signal processor platform for all imaging needs," *Electronic Imaging*, vol. 37, no. 7, pp. ISS–279, 2025.

[19] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *Proceedings of the International Conference on Quality of Multimedia Experience (QoMEX)*, 2016.

[20] M. Buckler, S. Jayasuriya, and A. Sampson, "Reconfiguring the imaging pipeline for computer vision," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 975–984.

[21] P. Hansen, A. Villegas, and R. Bregovic, "ISP4ML: The role of image signal processing in efficient deep learning vision systems," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2021.

[22] M. Yoshimura, J. Otsuka, A. Irie, and T. Yamashita, "DynamicISP: Dynamically controlled image signal processor for image recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[23] Y. Wang, Y. Liu, Y. Guo, Z. Wang, Y. Xing, W. Lian, and L. Ma, "AdaptiveISP: Learning an adaptive image signal processor for object detection," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[24] J. Chen, B. Wang, S. He, Q. Xing, X. Su, W. Liu, and G. Gao, "HISP: Heterogeneous image signal processor pipeline combining traditional and deep learning algorithms implemented on FPGA," *Electronics*, vol. 12, no. 16, p. 3525, 2023.

[25] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[26] C. Wu, J. Wang, and J. Yang, "VisionISP: Repurposing the image signal processor for computer vision applications," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2019.

[27] J. Hegarty, J. Brunhaver, Z. DeVito, J. Ragan-Kelley, N. Cohen, S. Bell, A. Vasilyev, M. Horowitz, and P. Hanrahan, "Darkroom: Compiling high-level image processing code into hardware pipelines," in *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 2014.

[28] A. Lopich, N. Yildiz, B. Dries, K. Griffiths, and B. Robson, "High-performance image signal processing and camera sensor pipeline design on FPGAs," Intel Corporation, Embedded Acceleration Division, White Paper WP-01337-1.1, Oct 2025. [Online]. Available: https://www.intel.com/content/www/us/en/content-details/827445/high-performance-image-signal-processing-and-camera-sensor-pipeline-design-on-fpgas-white-paper.html

[29] C. Poynton, *Digital Video and HDTV: Algorithms and Interfaces*. Morgan Kaufmann, 2003.

[30] I. Morawski, Y.-A. Chen, Y.-S. Lin, S. Dangi, K. He, and W. H. Hsu, "GenISP: Neural ISP for low-light machine cognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2022, pp. 630–639.

[31] V. Stimper, S. Bauer, R. Ernstorfer, B. Schölkopf, and R. P. Xian, "Multidimensional contrast limited adaptive histogram equalization," *IEEE Access*, vol. 7, pp. 165 437–165 447, 2019.

[32] T. Kuno, H. Sugiura, and N. Matoba, "A new automatic exposure system for digital still cameras," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 1, pp. 192–199, 1998.

[33] M. Nilsson, C. Weerasinghe, S. Lichman, Y. Shi, and I. Kharitonenko, "Design and implementation of a CMOS sensor based video camera incorporating a combined AWB/AEC module," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2, 2003, pp. 477–480.

[34] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda, "Photographic tone reproduction for digital images," in *Proceedings of SIGGRAPH*, 2002.

[35] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, "Contrastive learning for unpaired image-to-image translation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020, pp. 319–345.

[36] G. Jocher, A. Chaurasia, and J. Qiu, "YOLO by Ultralytics," https://github.com/ultralytics/ultralytics, 2023, version 8.0.

[37] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "SegFormer: Simple and efficient design for semantic segmentation with transformers," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021, pp. 12 077–12 090.

[38] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *International Journal of Robotics Research (IJRR)*, vol. 32, no. 11, pp. 1231–1237, 2013.

[39] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3213–3223.

[40] W. Snyder, "Verilator: Open-source Verilog simulation," https://www.veripool.org/verilator/, accessed: Jun. 13, 2026.

[41] S. Battiato, G. Messina, and A. Castorina, "Exposure correction for imaging devices: An overview," in *Single-Sensor Imaging: Methods and Applications for Digital Cameras*, R. Lukac, Ed. CRC Press, 2008.

[42] Swedish Parliament, "Camera surveillance act (kamerabevakningslagen)," https://www.riksdagen.se, 2018, sFS 2018:1200, with subsequent amendments.

[43] Integritetsskyddsmyndigheten, "Video surveillance and the general data protection regulation," https://www.imy.se/lagar--regler/dataskydd/gdpr/kamerabevakning, 2022, guidance on video recording in public spaces and applicability of the GDPR private use exemption.