FPGA Vision Systems for Autonomous Vehicles

Sangeetha Chandramouli
EECS221 – Advanced Computer Architecture
Department of Electrical Engineering and Computer Science
University of California, Irvine

***Abstract*— FPGA-Based Vision Systems for Autonomous Cars are at the forefront of cutting-edge autonomous vehicle technology, capitalizing on Field-Programmable FPGAs to process and analyse real-time data from a multitude of sensors, including cameras, LiDAR, and radar. This rapid data processing is the linchpin for essential functions like object detection, tracking, and decision-making, all of which are critical for the perception and control of autonomous vehicles. FPGA technology, celebrated for its low latency, energy efficiency, and parallel processing capabilities, is tailor-made to meet the data-intensive demands of autonomous driving. Research in this field encompasses a wide spectrum, including the development of robust hardware architectures, sophisticated algorithms, and efficient programming strategies to tackle the multifaceted challenges of real-time vision processing. The dynamic reconfigurability of FPGA-based systems allows them to adapt swiftly to shifting scenarios, enabling real-time tasks such as pedestrian and obstacle detection, multi-camera stereo vision, and precise object recognition. These capabilities, in turn, bolster the safety and reliability of autonomous vehicles navigating complex real-world environments. As the autonomous driving landscape continues to evolve at a rapid pace, FPGA-based vision systems stand as a cornerstone in enhancing the capabilities and real-world performance of self-driving cars. This paper focuses on FPGA integration with vision systems for automotive applications.**

***Keywords*— FPGA, Vision Systems, energy efficiency, self-driving cars**

I. PROBLEM STATEMENT

Achieving fully autonomous vehicles requires advanced on-board computer vision capabilities that far exceed the processing power of standard embedded systems. Safety-critical functions like pedestrian detection, lane tracking, and traffic sign recognition demand complex real-time image analysis algorithms. The ultra-high throughput needed for decision-making, along with stringent low latency requirements, present major computing challenges using traditional processors. This hinders progress towards fully self-driving systems.

This research article investigates FPGA platforms to address these compute-intensive vision workloads. With their inherently parallel architectures, FPGAs can efficiently execute the types of heavily parallelizable

vision tasks required. Various research is done on FPGA-based control system that integrates lane detection, traffic light recognition, obstacle detection and pedestrian identification to enact navigation decisions. They also survey on FPGA implementations for stereo image depth mapping, a lower cost but computationally demanding alternative to LIDAR depth sensing. All those techniques optimize performance using FPGA-tailored strategies like pipelining, data parallelism and custom memory structures. Figure 1[1] represents the full functionality of an autonomous vehicle.

S.NO	Acronym	Description
1	FPGA	Field-Programmable Gate Arrays
2	CPU	Central Processing Unit
3	GPU	Graphics Processing Unit
4	SAE	Society of American Engineers
5	CNN	Convolutional Neural Network
6	NN	Neural Network
7	TSR	Traffic Sign Recognition
8	SVM	Support Vector machine
9	DL	Deep Learning
10	HLS	High-level Synthesis
11	IoU	Intersection over Union
12	V2X	Vehicle-to-everything
13	SGM	Semi-Global Matching
14	ADAS	Advanced Driving Assisted Systems
15	HOG	Histogram of Oriented Gradients
16	RMSE	Root Mean Squared Error
17	mAP	mean Average Precision
18	PCA	Principal Component Analysis

Table 1: List of Abbreviations

II. INTRODUCTION

FPGAs have emerged as an attractive technology for accelerating computer vision algorithms in autonomous vehicles. They provide high performance and energy efficiency compared to CPUs and GPUs for the latency-critical vision processing tasks crucial for self-driving cars. Techniques like pipelining, parallelism, and configurable memories can be exploited to optimize performance. The power, latency, and accuracy requirements for on-board systems in autonomous vehicles are also noted. Estimations suggest that around 100 TFLOPS[2] of computing power may be needed to run all vision algorithms in real-time at 30 frames per second. Meeting such high performance at low power at low budgets is crucial but challenging.

The unique feature of FPGA makes them well-suited to meet these demands. Their spatial architecture and inherent parallelism can be exploited to accelerate computer vision workloads. Techniques like pipelining, parallel processing and custom memories can enable optimizations not feasible on CPUs and GPUs. FPGAs also offer higher energy efficiency stemming from customized precision and data flows.

Detailed empirical comparisons between FPGA and non-FPGA solutions are presented for metrics like accuracy, throughput and energy efficiency. Results reveal that FPGA throughput is comparable to or exceeds that of GPU alternatives in certain cases owing to spatial advantages. However, accuracy gaps persist in safety-critical functions like pedestrian detection which is a major on-going challenge. The higher accuracy of models

on CPUs/ GPUs stems from greater flexibility in handling large neutral networks.

Another persistent challenge includes reducing end-to-end latency to meet hard real-time constraints and enhancing productivity of FPGA programming tools like HLS for wider adoption. The detailed state-of-art overview provides a valuable guide for researchers venturing FPGA-based solutions in this societally impactful emerging domain.

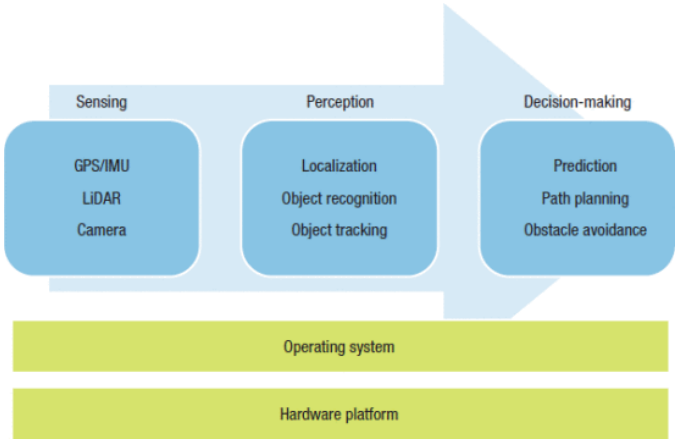

Figure 1: Tasks in autonomous driving

Figure 2 represents the different levels of autonomous driving defined by SAE. At Level 0, the driver performs all driving process while level 5 entails full autonomy with the autonomous system carrying out all driving functions under all conditions. The intermediate levels denote gradual transitions where certain driving modes are handled by the vehicle, but human override maybe needed in certain complex scenarios. For instance, at level 3, environmental monitoring is achieved by the automated system, but the human driver must be receptive to requests to intervene.

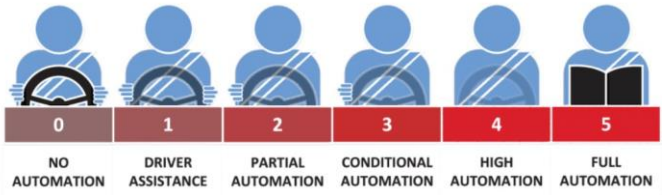

Figure 2: Levels of autonomous driving as defined by SAE

III.FPGA TECHNOLOGY IN AUTNOMOUS VEHICLES

Developing fully autonomous self-driving vehicles requires sophisticated environmental perception and navigation capabilities to replace human drivers. These demands processing sensory data from cameras, radars and lidars to identify lanes, traffic elements, obstacles, free space around the vehicle and estimating precise location and movement of the car. Sensor inputs feed algorithms that dynamically model the surroundings and chart out safe trajectories to destination avoiding collisions. Figure 3 shows the Main Control Architecture of the design. The main controller manages the navigation task. It receives input from the detection modules. These modules provide control signals in order to indicate the function detected[3].

Attaining reliable autonomous functionality requires tremendous computing of throughput - nearly 200 TOPS for sensor processing and over 1 TOPS[4] for planning and

control routines per recent benchmarks. Performing such data and compute intensive processing within milliseconds necessary for real-time response mandates specialized computing machinery.

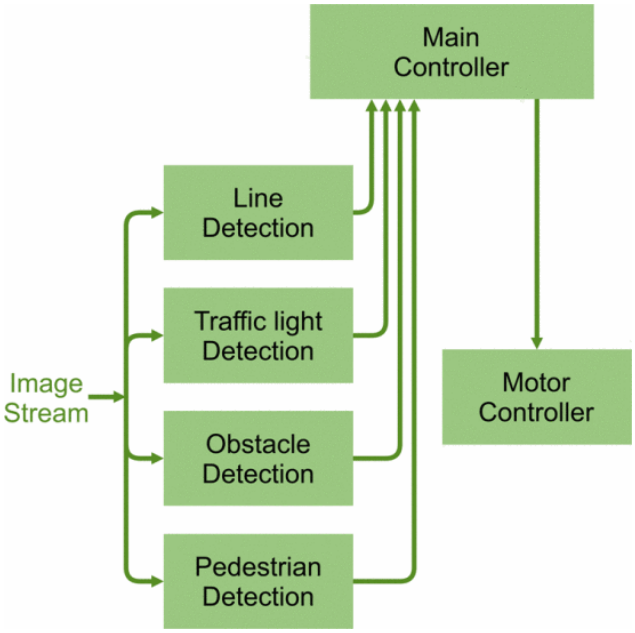

Figure 3: Main Control Architecture

Mainstream serial processors in today's cars lack the expected throughput and energy efficiency for autonomy. Massively parallel solutions like FPGA, tuned to embedded vision and control workloads are necessary instead. FPGAs provide a flexible fabric of reconfigurable logic and memory blocks interconnectable to specific applications. Multiple replicated computing units unlock substantial parallelism in perception tasks to greatly boost performance compared to traditional serial processing.

For example, detecting objects using a CNN requires sliding a boundary box over input images and running classifiers over numerous sub-regions concurrently. This can be realized on FPGAs by unfolding classifier logic blocks to gaze at different image tiles simultaneously. Stereo vision estimates scene depth by finding pixel matches between left and right camera images. Searching for these disparities in parallel accelerates this process. Matrix multiplications dominate modern convolutional networks used for recognition and segmentation. Special processing arrays like systolic architectures avoid repeatedly fetching weights thereby accelerating such operations. The distributed RAM banks prevent frequent data transfers while reduced bit widths minimize resource usage without significantly impacting output quality. Algorithms are split into deeply pipelined stages allowing different vision pipeline blocks to process separate batches concurrently as new image pixels stream in. Together, these facilitate high throughput - hundreds of FPS needed for real-time performance even on computationally heavy 4K video feeds.

For planning, the FPGA logic readily encodes behavior trees governing discrete maneuver decisions based on environmental awareness. Adaptable device interfaces simplify integrating sensors for reliable perception and wiring actuator control command streams. High level path finding formulations leverage integrated multicore CPUs. Fail-operational needs are met via replicating

critical modules and voting between outputs. Synchronous design principles aid noise resilience.

Compared to generalized software-driven alternatives FPGA solutions boast 10X better power efficiency[5], vital in battery-powered vehicles. Small form factor aids placement on space-constrained on-board computers. Quick reprogramming permits runtime tuning of capabilities responding to scenario changes. The customization flexibility however compromises utilization next to fixed-function ASICs optimized for one model alone. But FPGAs provide necessary agility to handle the rapid evolution ongoing in self-driving research and requirements.

Sustained progress in design tools, libraries and architectures to abstract away low-level complexity will further bolster FPGA adoption. Overall, FPGAs deliver the right balance of performance, efficiency and flexibility crucial for developing and deploying autonomous driving tech. Their automotive-grade variants with functional safety and thermal guarantees suit the stringent operating conditions faced by algorithms guiding future driverless cars[6].

Modern FPGAs integrate heterogeneous components like multi-core ARM processors for software programmability, high bandwidth external memory interfaces (HBM, DDR, HMC) for data intensive workloads and high-speed connectivity like PCIe, Ethernet for device communication. State of the art toolchains incorporate High Level Synthesis so C/C++/OpenCL code can be compiled to optimized circuits by extracting parallelism and concurrency intrinsic to the algorithms thereby abstracting low-level RTL coding complexities from developers.

Leveraging these aids and inbuilt architectural enhancements, FPGAs have proven capable to deliver the key autonomous driving competencies required:

1. High frame rate sensor analytics performing object detection, semantic segmentation, traffic sign recognition etc. at 30+ FPS under varying lighting.
2. Depth image generation from stereo cameras via FPGA accelerated block matching and cost aggregation techniques.
3. NN inference for perception, intent analysis etc. using Systolic Array based Tensor Processing Units.
4. Sensor input fusion and world modeling combining camera, radar, lidar streams via array Kalman/Particle filtering.
5. Route planning solving optimization objectives around waypoint selection, trajectory generation etc.

6. Rapid actuator control by direct wiring to command vehicle mobility and teleoperation.
7. Hardware redundancy using partial reconfigurability for guaranteeing fault tolerance.

The fine-grained flexibility balances customization for efficiency yet retains generalization needed to handle evolving autonomous driving research advancements. By exploiting their intrinsic spatial architecture and capacity for application specific optimization, FPGAs deliver a high performance, adaptable heterogeneous platform to progress self-driving car development.

A. Advantages of FPGA

- Performance - The parallel architecture of FPGAs enables accelerating compute-intensive automotive workloads like ADAS vision processing and autonomous driving algorithms. Their pipelined execution helps meet the real-time constraints[7].
- Power Efficiency - Customized data flows and reduced numerical precision led to significant power savings versus software-based implementations. This extends electric vehicle range.
- Reconfigurability - In-field updates to ADAS and infotainment algorithms are supported by dynamically reprogramming FPGAs. New functionalities can be added post-deployment.
- Functional Safety - Integrated redundancy and lock-step cores allow safety certification per ISO26262 ASIL-D. Critical fail-safe requirements for autonomous systems are addressable.
- Small Form Factor - The compact, thermally rugged package facilitates embedding FPGAs flexibly within space-constrained on-vehicle domains alongside custom chips and processors.

IV. IMPORTANCE OF COMPUTER VISION IN AUTONOMOUS VEHICLE

The capabilities of autonomous vehicles heavily rely on advanced computer vision systems to perceive and properly understand the vehicle's surroundings in real-time. Without sophisticated computer vision algorithms and techniques that can interpret imagery and sensor data, self-driving cars would not be able to safely navigate and operate on roads.

Object detection enables identification and classification of entities like other vehicles, pedestrians, roads signs, obstacles, and lane markings by processing sensor feeds from cameras, LiDAR, radar and more. Real-time detection provides fundamental environmental awareness

that is crucial for automated driving. Deep NN like CNN powered by computational advances in GPU hardware now allow accurate detection capabilities. Algorithms like R-CNN, SSD, YOLO and derivations like Fast R-CNN can detect multiple object types reliably. Vehicle-mounted multi-camera systems with wide angles and zoom, covering diverse viewing angles around cars are what feed imagery to the advanced detection algorithms. Failovers across such multi-camera installations account for any misses, ensuring robust operation.

- R-CNN: Regions with CNN features applied to generate region proposals and then classify using CNN. Improved versions are Fast R-CNN and Faster R-CNN.
- SSD: Single Shot Detector that uses a single feedforward CNN to directly predict classes and anchor offsets without region proposals.
- YOLO: You Only Look Once also uses a single CNN but divides image into grid and predicts bounding boxes and class probabilities for each grid cell.

Once objects are detected accurately, estimating precise motions and tracking them over time allows prediction of future locations and trajectories - key to planning safe and reactive driving strategies. Tracking techniques like kernelized correlation filters, frequently combined with Kalman filters as well as recurrent networks like long short-term memory networks enable reliable real-time tracking and motion estimation. Measurements from onboard vehicle motion sensors also get integrated to account for ego motion of the self-driving vehicle itself. The output of tracking systems feeds into navigation metrics, route and motion planning components that connect to actual vehicle control interfaces. Some of the tracking algorithms are:

1. Kernelized Correlation Filters: Use Gaussian kernel to approximate discrete impulse response for tracking.
2. Kalman Filter: Estimates state of a linear dynamic system using measurements over time. Commonly used to track position and velocity. Tracking equation:

$$X^p_t = AX_{t-1} + Bu_t + w_t$$

where X_t is the state vector at time step t, A is the state transition model, B is the control input model, u is the control vector, and w is process noise (error terms).

Mercedes Benz notes that over one terabyte of visual data can be processed per hour by on-vehicle computers that power autonomy in their vehicles - stressing the real-time processing demands of computer vision. The entire pipeline needs efficient algorithms - lane detection functionality leverages Hough transforms to find road markings for example, while traffic light recognition relies on circle boundary analysis. Reading signs uses

feature extraction techniques like SIFT and ORB coupled with multiclass classifiers to read speed limits, stop signs and more. Specific techniques handle particular sub-problems in machine perception for self-driving.

In Depth Estimation (Stereo Vision), the disparity d is the difference in x-coordinate between matched points in left and right camera images. Depth Z is inversely proportional to disparity as shown in figure 4:

$$Z(i, j) = Bf / d(i, j)$$

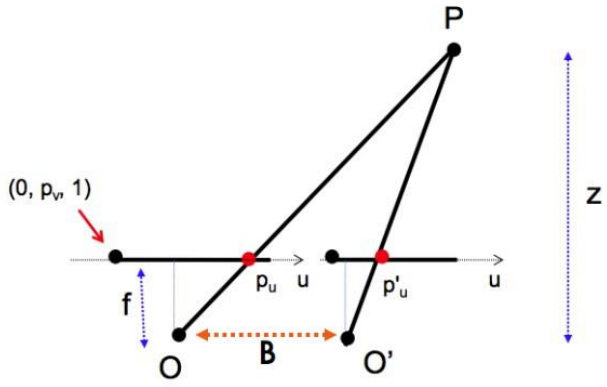

Figure 4: Computing depth

where B is distance between cameras, f is focal length.

There are still challenges faced by even state-of-the-art autonomous vehicle computer vision techniques - like detecting pedestrians accurately in low light or safely handling left turns in dense cross-traffic situations. Strategies like using convolutional nets modelled on biological vision to work across contexts show promise to handle such issues. Continual tuning of algorithms using growing datasets from increasing test miles also continues, as systematic testing and failure analysis processes mature in the autonomous vehicle industry. But despite hurdles, computer vision has enabled remarkable automated perception capabilities - and continues to be the most critical technology for making self-driving.

V. EVALUATION METRICS

Evaluating performance of algorithms that enable autonomous functionality requires rigorous quantitative metrics assessing correctness, safety, and reliability across all facets of self-driving pipelines. Key criteria span verification of perception competence, motion planning soundness, actuator control responsiveness and failure tolerance guarantee.

1. Perception Accuracy: Environment sensing is foundationally vital for downstream driving decisions. Vision perception tasks leverage camera and lidar streams to understand semantics and geometry of surrounding regions. Semantic segmentation using CNN classifies pixels into road, vehicles, pedestrians etc.

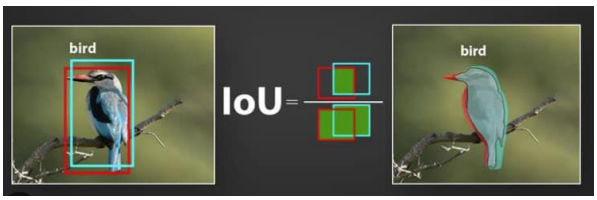

Figure 5: IoU for Object Detection

The IoU metric **(**Eq. 1**)** evaluates semantic classification accuracy by gauging region overlap between predicted segmentation and ground truth labels. Higher IoU indicates precise delineation of road objects.

$$\text{IoU}= \frac{TruePositives}{TruePositives+FalsePositives+FalseNegatives} \quad - 1$$

Object detectors like YOLO, SSD leverage boundary box regression to localize entities. Along with class-specific localization precision, mAP averaged over C detectable classes **(**Eq. 2**)** evaluates multi-class recognition accuracy.

$$\text{mAP} = \frac{1}{n}\sum_{k=1}^{n} AP_k \quad - 2$$

where AP_k is the average precision for class k. Algorithms estimating scene depth from stereo vision perform sparse or dense matching of left-right images. For autonomous navigation, position and scale understanding must be metrically accurate. Error in inferred range or disparities is measured via RMSE between predicted and LIDAR ground truth depth, disparities or surface normal values (Eq. 3). Lower RMSE indicates precision.

$$\text{RMSE} = \sqrt{(\frac{1}{N}\sum_{i\epsilon N}(\widehat{y_i}\text{-}y_i)^2)} \text{ - } 3$$

2. Localization and Mapping Exactness: Precisely estimating the vehicle's dynamic state (pose, velocity) by continually tracking features detected in camera and lidar streams is fundamental for localization. Algorithm robustness and drift over distance travelled quantifies consistency. Prominent techniques like iterative closest point matching, bundle adjustment (Eq. 4) minimize reconstruction loss between sensed surfaces, landmark maps to estimate evolving placement in 6-DOF pose space.

$$min_{a_j,b_i} \sum_{i=1}^{n}\sum_{j=1}^{m} v_{ij}\text{d}(\boldsymbol{Q}(a_j, b_i),\ x_{ij})^2 - 4$$

where $Q(a_j, b_i)$ is the predicted projection of point i on the image j and d(x, y) denotes the Euclidean distance between the image points represented by vectors x and y.

3. Planning Fidelity: Route planners leverage perceived environmental maps and constraints to chart collision-free paths balancing progress, safety and ride comfort. Evaluation relies on empirically scored metrics measuring closeness to lane centres, curvature smoothness, adherence to traffic conventions, unimpeded velocity etc. over entire trajectories. Replanning frequency given perception changes assesses adaptiveness. For interactive scenarios like unprotected turns, merging, crossing etc. interactivity models score individual behaviours and cohort interactions over thousands of test cases simulating posing vehicles, pedestrians for statistical verification of rational, safe driving conventions. These complement dynamic real-world validation.

4. Control Responsiveness: The velocity and steering actuation response lag from sensed stimuli determines dynamic manoeuvrability. Tight control loops are vital for highway speeds. Control latency (Eq. 5) must be verified to sub-100 millisecond levels dictated by 60 mph velocities. Pipeline stages spanning sensor capture, perception processing, behaviour decision and mechanical actuation are profiled to minimize lag.

$$Latency = SensorCapturing + PerceptionProcessing + DecisionResponse + ActuationLag \quad - 5$$

5. Functional Safety: With autonomy software governing pedestrian lives, functional safety standards mandate stringent diligence. Potential failure modes are exhaustively identified using techniques like Fault/Event Tree Analysis decomposing harmful system states down to component faults. Safety mechanisms then institute redundancy in critical modules, sensor/compute channel isolation, runtime monitors, degraded operational modes etc. to guarantee risk mitigation to acceptable levels.

Across these facets spanning perception, planning, control and safety, test metrics quantify confidence in the autonomous vehicle solution. Standardized public benchmarks facilitate development while high fidelity simulators synthesizing billions of operational miles expose corner cases aiding verification. But cautious incremental deployment in real-world driving accumulates safety assurance towards validated performance under generalized operational design domains.

VI. APPLICATIONS

A. *LANE DETECTION*

Lane detection is an important function for ADAS and autonomous vehicles. It detects and locates lane markings on the road to provide critical information for lane departure warning systems, adaptive cruise control, lane keeping assist, etc. Figure 6 represents the flow of lane detection[8].

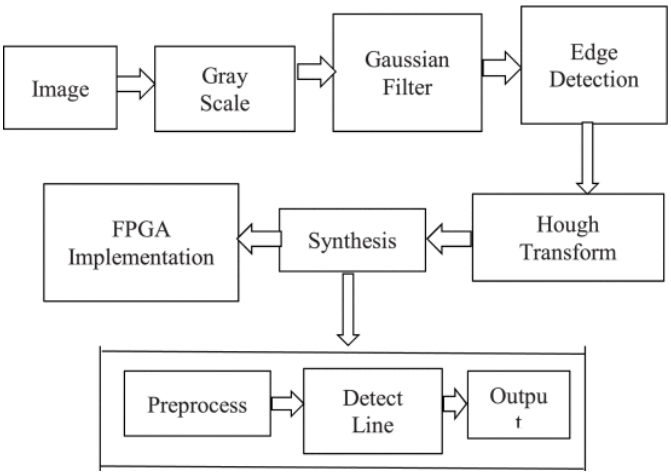

Figure 6: Block Diagram of Lane Detection

The lane detection algorithm typically consists of several steps: image acquisition, pre-processing, edge detection, region of interest (ROI) selection, lane model fitting, and tracking. The input to the lane detection system is a video

stream acquired from one or more cameras mounted on the vehicle. Typical resolutions range from 640x480 VGA to 1080p or higher. Low light sensitivity and auto exposure controls allow the cameras to adapt to varying lighting conditions. Images are captured at a high frame rate, generally 30 fps or faster, to detect lanes with minimal latency as the vehicle moves rapidly along the road. Common pre-processing includes grayscale conversion and Gaussian filtering to reduce noise. The raw RGB data is converted to grayscale imagery using linear combinations of the R, G and B channels (e.g., 0.2989 R + 0.5870 G + 0.1140 B). This simplifies subsequent processing yet retains the luminance detail that captures lane markings. A low pass Gaussian filter is applied to suppress high frequency noise. This smoothing retains structures and edges while removing unwanted camera sensor noise.

Only the bottom half of the image is retained for further processing. This focuses computations on the road area immediately ahead of the vehicle containing the desired lane markings. Lane markings appear as strong edges in the road scene imagery. Popular edge detection methods are Canny, Sobel, and Prewitt which identify pixels with intensity discontinuities. The Hough Transform is then applied to fit lane models based on the edge pixels. The Hough Transform is applied to the edge detected image to detect straight line segments that may correspond to lane boundaries. It converts image space into Hough parameter space where lines can be efficiently aggregated. Tracking refines the lanes using temporal information with Kalman Filters.

On FPGAs, processing can be accelerated by unrolling loops and replicating processing units. For example, multiple Hough Transform units can vote in parallel to speed up line detection. On-chip dual port RAMs allow simultaneous read and write access to avoid collisions. Lookup tables are used instead of complex computations like sine/cosine. Dataflow optimization techniques like line buffers, window buffers, and ping-pong buffers increase data locality and reuse while reducing external memory bandwidth. Arithmetic operations use optimized fixed-point representation.

FPGAs provide flexible, low latency and power efficient hardware fabrics to implement lane detection pipelines.

The vision algorithms are coded in a hardware description language (HDL) like VHDL or Verilog or high-level languages (HLL) like C/C++. These are synthesized into gate level netlists mapped onto FPGA logic blocks and DSP slices using CAD tools. Key optimizations include:

i. Pipelining: Breaks operations across sequential stages to enable concurrent,

parallel execution across multiple image pixels.

ii. Custom Memory Buffers: Small local storage like shift registers and FIFO buffers eliminate redundant external memory accesses.

iii. Loop Unrolling: Replicates computational hardware like Hough voting units to operate on separate data chunks improving throughput.

iv. Fixed Point Arithmetic: Uses finite bit widths vs floating point to save logic resources while meeting accuracy needs.

These approaches boost performance to achieve real-time frame processing requirements. The lane detection outputs can directly control vehicle steering or feed into more complex autonomous driving algorithms.

A sample lane detection algorithm is implemented on Xilinx Zynq 7000 using HLS synthesis, achieving 126 FPS on 64x64 images with 88ms max processing time per frame. Onchip BRAM usage is optimized to support larger images. Logic implementation utilizes Gaussian filter and Canny edge detection for robust lanes detection. Hough transform fits lane models which are stored in BRAMs.

Evaluation metrics for lane detection accuracy include precision, recall, F1 score, and accuracy. Precision is percentage of detected lanes that are true lanes. Recall is percentage of actual lanes that are correctly detected. F1 score combines precision and recall as their harmonic mean. Accuracy measures how often lanes are correctly identified. For real-time testing without ground truth data, true positives, false positives, false negatives are used.

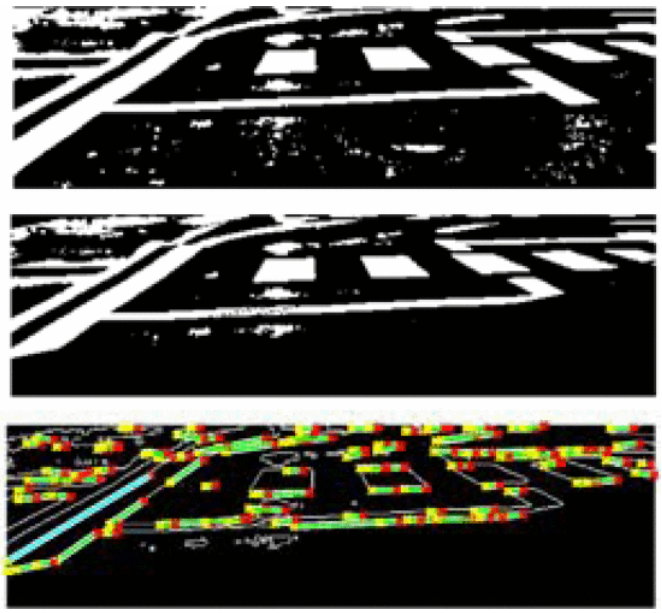

Figure 7: Lane Detection. Top: Binary Image; Middle: Filtered image; Bottom: Hough transform result

In terms of performance, throughputs of up to 60 FPS are reported for 1280x720 images. For accuracy, existing FPGA works are mainly based on Hough transforms which limit maximum accuracy. There is a gap versus advanced software solutions, especially using neural networks, which achieve higher accuracy already at real-time rates on CPUs/GPUs. Regarding energy efficiency, estimates show 2-58x improvements versus GPUs. However, analyses overlook critical aspects like arithmetic precision. Still FPGA solutions overall provide

better energy efficiency, but more thorough evaluations are required.

FPGAs allow optimized lane detection implementations with real-time performance. But there are gaps in accuracy and validation versus state-of-the-art software solutions. Evaluations overlook critical energy efficiency factors as well. Main open challenges include closing accuracy gap with NN suited for FPGAs, more rigorous benchmarking on public datasets, and models addressing metrics like collisions avoided.

B. TRAFFIC SIGN RECOGNITION

TSR enables vehicles to detect and classify traffic signs, interpreting key information about road conditions, vehicle control and safety. This capability is essential for autonomous and advanced driver assistance systems. TSR typically comprises two phases - detecting signs in images from cameras, and then classifying their meaning. A common approach for detection is to first segment probable sign locations based on color and shape attributes[9]. Signs have standardized color schemes detectable in HSV color space. Morphological smoothing and Canny edge detection then finds shape contours. Circular Hough transforms can identify segment boundaries, representing circles parametrically through voting. Key formulas include HSV color conversion aligning perceptual channels, and the circular Hough transform using radius and centre point coordinates. FPGA implementations accelerate Hough voting via parallel hardware lanes. Key aspects is converting RGB images to HSV color space for perceptual uniformity, applying erosion/dilation to smooth shapes, Canny edge detection to outline contours, and finally Hough circle voting to parametrically fit circular boundaries. Figure 8 shows the detection process of color space algorithm[9].

Figure 8: Color Space Detection

With locations and segments of likely signs obtained, the next phase recognizes their semantic meaning. This relies on distinguishing features resilient to viewing angle, distance and lighting variations. PCA reduces feature dimensionality. PCA finds the most informative linear combinations of inputs for compact representation. Zernike or Hu invariant moments[10] capture essential shape cues. A SVM then categorizes sign types using these rotation and scale invariant signatures. SVMs classify based on a maximum margin hyperplane decision boundary in higher-dimensional space. SVMs mainly separate classes via maximum margin hyperplanes in higher-dimensional space. The boundary is learned from labelled training examples. Alternatively, NN stack

multiple layers of hierarchical visual features, learning complex patterns from data. NN thus adapt better but require more training.

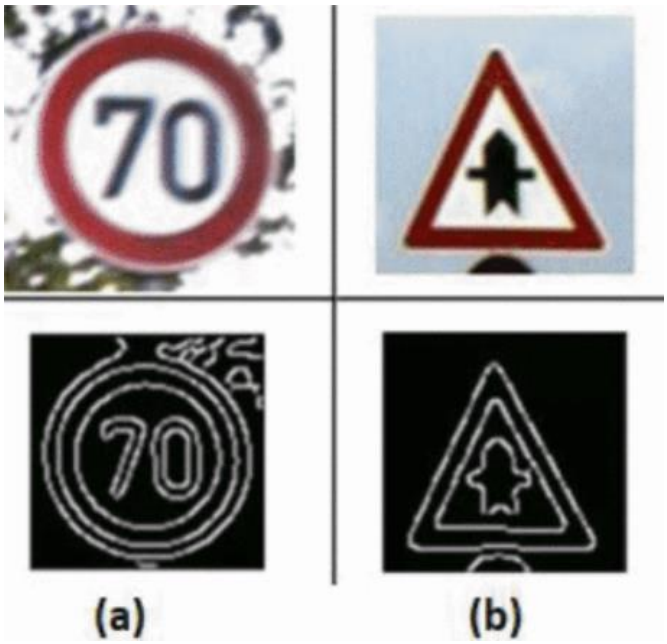

Figure 9: Traffic Sign Edge Detection[11]

Overall, FPGAs confer significant advantages for embedded TSR systems compared to CPU or GPU execution. Flexible parallelism allows customized deep pipelines and processing arrays tuned to algorithms. Local memories such as line buffers minimize data transfers. Fixed-point math saves logic resources and power. Loop unrolling and systolic architectures further boost math throughput. Together these optimization mechanisms enable accurate, low-latency TSR amenable for automotive integration. Future work should focus on robust operation despite lighting, weather, or sign variations, while interfacing TSR to downstream autonomous functionality like planning and control.

The combination of computer vision techniques with tailored FPGA acceleration thus provides an efficient foundation for TSR in emerging self-driving systems[11]. As algorithms and datasets mature to handle diverse real-world driving scenarios, TSR will become a standard capability enhancing safety and mobility. The flexibility of modern FPGAs allows cost-effective upgrades to evolving TSR subsystems. Together with complementary sensors, ever-improving perception will move autonomous vehicles closer to widespread adoption. Figure 9 represents the SVM identification of logo by using extracting Hu and Zernike moment invariant features.

C. PEDASTRIAN DETECTION

Ensuring pedestrian safety is a pivotal requirement for autonomous vehicles. Pedestrian detection systems based on computer vision techniques can analyse images captured by cameras in real-time to identify people on or around roads. However, high-resolution image processing involves intensive computations that demand optimized hardware like FPGAs for real-time performance.

A practical pedestrian detection pipeline first extracts feature from input images that can distinguish human shapes and forms. The HOG algorithm[12] as shown in Figure 10, is widely used here. It divides images into small windows, computes gradient information, and

aggregates this into histograms that capture dominant edge orientations – useful to identify pedestrians. Other techniques like Haar wavelets that encode shape profiles are also options.

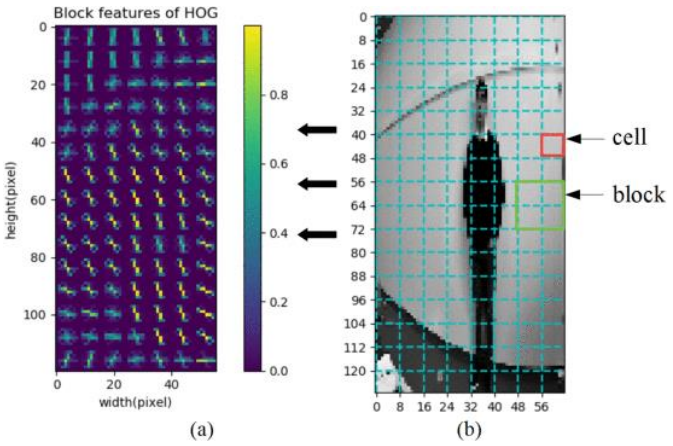

Figure 10: HOG Feature

With feature vectors generated, the next stage is to classify windows as pedestrian or non-pedestrian via machine learning models. Algorithms like SVMs, AdaBoost, and recently deep NN provide high accuracy here. Hardware-friendly SVMs classify data by mapping inputs to a high-dimensional space and identifying boundaries between classes. They work well with HOG features.

However, DL methods like CNN now surpass classical techniques, achieving over 98% accuracy on pedestrian benchmarks by learning hierarchical, spatial feature representations. State-of-the-art networks like Mask R-CNN use a two-stage approach: proposal generation to first identify pedestrian locations, followed by bounding box adjustments and segmentation to isolate the pedestrians. Such frameworks provide incredible accuracy but involve billions of compute operations.

This makes FPGA-based hardware acceleration essential for real-time performance. FPGAs like Zynq UltraScale+ provide reconfigurable logic fabrics to implement custom architectures optimized for algorithms like HOG. Using HLS from C/C++, designers can rapidly prototype systems with techniques like pipelining, parallelization and data streaming to boost performance. On-chip memories also eliminate expensive external memory traffic. Together, this achieves around 10X higher FPS over CPUs for pedestrian detection with HOG+SVM.

However, accuracy currently suffers on FPGAs relative to GPUs or advanced driver assistance platforms using DL. Quantization, pruning, and network simplification help port large CNN, but constrained FPGA resources limit model complexity versus GPUs. Hybrid CPU-FPGA computing is an option, using FPGAs to accelerate parts of pipelines. Ultimately, optimized FPGA-based CNN that balance accuracy, speed and efficiency remain an open research challenge. Enhancing high-level design tools is also important so non-hardware experts can effectively implement solutions.

FPGAs deliver substantial real-time performance gains and power efficiency benefits for pedestrian detection algorithms, but accuracy lags state-of-the-art techniques.

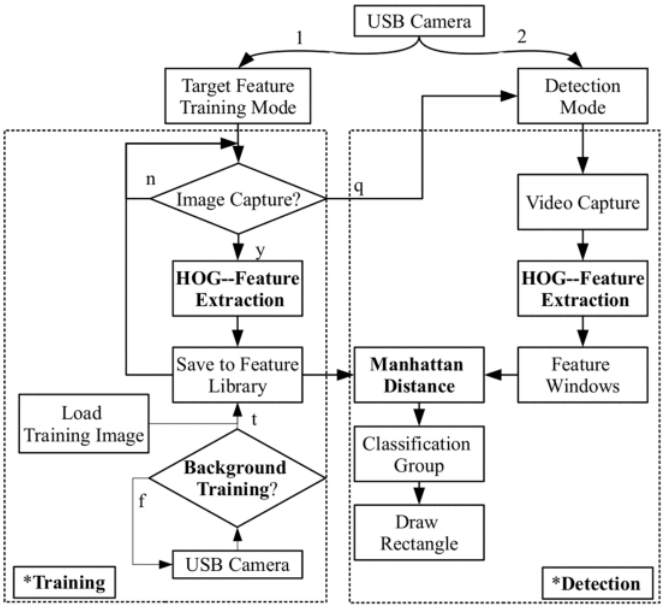

Figure 11: Pedestrian Detection System

As autonomous vehicles advance, boosting accuracy without sacrificing efficiency will be crucial using co-design approaches spanning algorithm innovation, software frameworks and specialized hardware like FPGAs. This can translate the rapid progress in computer vision into next-generation systems that reliably detect pedestrians, ensuring public trust and safety. Summary of pedestrian detection process is shown in Figure 11.

D. DEPTH ESTIMATION

Depth estimation is a key capability required for safe autonomous driving to understand the vehicle's surrounding environment. While other sensors like LIDAR can provide depth information, estimating depth from cameras using computer vision techniques is a lower cost approach that leverages existing visual sensors. FPGAs offer an efficient platform to implement these vision algorithms compared to CPUs and GPUs.

The core idea in depth from stereo vision is that given two images captured from cameras displaced horizontally, the difference in position of the projection of objects at different depths allows inferring their distance. This difference between the projections is called disparity. Using the focal length and baseline distance between cameras, disparity maps can be converted to per-pixel depth estimates[13]. Computing correspondences between the stereo images is challenging due to aspects like varying illumination, lack of texture, occlusion etc.

SGM is a popular stereo matching algorithm for real-time applications[14]. It converts matching to an energy minimization problem. FPGAs implementations use pipelining and parallelism optimally to accelerate SGM. The high memory bandwidth enables fast cost computation and aggregation along different directions by efficient buffering. Techniques like loop unrolling, fixed point arithmetic, and minimizing memory accesses are employed for an efficient design. Recent works have added NN to improve matching costs. With optimization, FPGA implementations using simplified SGM have demonstrated high throughput of 100+ FPS for VGA resolution at under 5W power consumption.

Figure 12: Disparity Image computed using SGM

Figure 12[14] depicts an example of depth estimation by matching regions of the image from one camera to another. Both cameras have images from two viewpoints of same scene.

While less accurate, monocular depth sensing is an active area of research using DL methods like regression and classification CNN. FPGA CNN accelerators face challenges due to limited on-chip memory and optimizing for varied layer configurations. Quantization, pruning and reduced precision arithmetic help fit larger models but limit accuracy. Still, throughput over 30 FPS has been shown for certain architectures. Hybrid CPU-FPGA systems can allow executing larger models by apportioning computation.

FPGA based solutions have demonstrated promising results for real-time depth estimation from visual sensors. Leveraging inherent parallelism and pipelining capabilities, optimized implementations of algorithms like SGM and CNN can meet speed, power and accuracy trade-offs suitable for self-driving vehicles. Integrating multiple data streams with supplemental techniques can further improve the perception reliability necessary for safe operation. Recent datasets captured specifically in driving scenarios will help benchmark progress too. Overall, FPGAs complement the AV compute stack as efficient coprocessors for such latency-critical vision tasks.

VII. EXPERIMENT AND RESULTS

An FPGA-based control system to autonomously drive a vehicle was designed[3]. A camera mounted on the vehicle along with a custom FPGA board was used to implement real-time image processing algorithms and vehicle control as shown in Figure 13.

Key algorithms they developed included lane detection, traffic light detection, obstacle detection, and pedestrian detection. For lane detection, a pipeline consisting of guided filtering was used for noise reduction, Otsu thresholding to convert the image to binary, morphological filtering to remove small noise areas, and Hough transforms to detect lane lines. This enabled

identifying the road edges, stop lines at intersections, and zebra crossings.

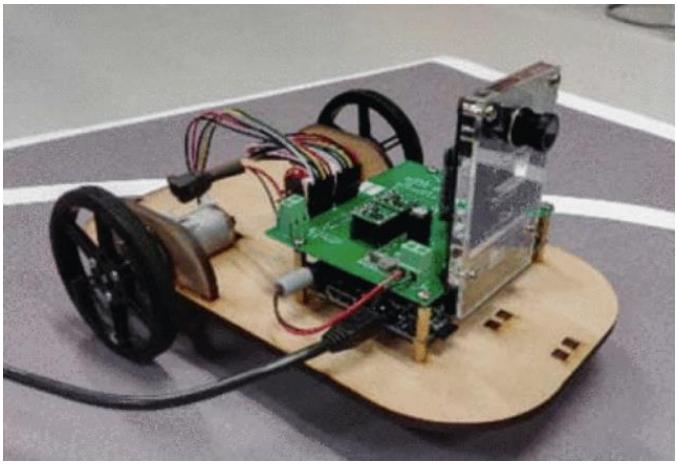

Figure 13: Experimental Setup

For traffic light detection, the image was converted to HSV color space and color thresholding tuned to the competition lights' red, yellow and green colors. By detecting the brightest light region, light that got lit was determined. For obstacle and pedestrian detection, again HSV color space was used and morphological filtering techniques to isolate foreground objects on the road from the background. Additionally for pedestrians, a bounding box was fitted and checked proportions to identify the torso and confirm human shapes.

The overall control architecture used hierarchical state machines to integrate these vision algorithms and make appropriate navigation decisions. For example, detecting a stop line would trigger states for slowing down and stopping, while a green traffic light would allow continuing. Lower-level motor controllers with PID feedback were used to realize the navigation commands.

Initial development and testing of the algorithms was done in MATLAB. This allowed verifying the image processing techniques before porting the MATLAB code to VHDL code to run on the FPGA.

VIII. CONCLUSION

In closing, this paper provided a comprehensive overview of FPGA-based vision systems for autonomous vehicles. We discussed how the inherent parallelism and pipelining capabilities of FPGAs can be leveraged to accelerate critical perception algorithms for self-driving cars. Coupled with customized bit widths and data flows, FPGAs achieve orders of magnitude better power efficiency versus GPU alternatives that software frameworks rely on presently. We surveyed relevant research across major vision tasks - from pedestrian detection to traffic sign recognition. In each case, strategies like loop unrolling, local memory structures and fixed-point arithmetic extract performance speedups of 10-100X over CPUs while retaining accuracy[1].

However, the are some noted gaps versus state-of-the-art techniques, especially advanced neural networks only hostable on GPUs now. Enabling such complex models on FPGAs through improvements in programmability and device utilization is an open challenge. We also outlined

deficiencies around standardized benchmarking using traffic datasets and verification of complete self-driving pipelines. Overall, FPGAs confer strong potential to complement AV system design via their versatility straddling hardware real-time guarantees and software productivity. Sustained progress in architectures, tools and frameworks will expand their accessibility to non-experts. Together with sensor fusion and safety mechanisms, FPGAs promise to bolster realization of robust, trustworthy autonomous driving capabilities.

IX. FUTURE SCOPE

The integration of FPGAs and computer vision systems into autonomous vehicles holds great promise for enhancing automotive safety and driving quality. However, current research efforts still face several key shortcomings that need to be addressed before this technology reaches widespread adoption.

One major issue is that existing FPGA and computer vision algorithms for autonomous driving tend to be computationally intensive and have high power demands. This strains onboard processors and reduces overall vehicle efficiency and range per charge for electric vehicles. More optimized FPGA code and efficient computer vision models like MobileNets[3] that retain accuracy while requiring fewer operations could help mitigate these problems.

Additionally, current algorithms are still not robust enough to handle the long tail of rare or corner driving scenarios. This includes challenges like navigating unexpected construction obstacles, handling inclement weather, understanding hand signals from police officers directing traffic, and appropriately reacting to emergencies like crashes or debris in the roadway. Expanding training datasets and leveraging generative adversarial networks to synthesize additional unusual driving scenarios could produce more adaptive systems.

Testing and validation of FPGA reliability poses another barrier for autos. The automotive industry requires far more stringent standards and test procedures before integrating new hardware than consumer electronics. Hardening FPGA tools and workflows to capture logic errors, guarantee fault tolerance, and demonstrate functional safety will be critical for adoption. This includes detailed documentation and proof that the FPGA adequately restricts system failures that lead to injuries or death.

Looking ahead, an important frontier for FPGA and computer vision innovation is V2X communication for collaborative smart transit. There is enormous potential to share sensor data and driving intentions between nearby vehicles, traffic signals, and infrastructure to coordinate movements. However, this will necessitate ultra-low

latency responses not consistently achievable today, demanding next-generation high-speed communication interfaces integrated tightly with FPGA logic.

Likewise, enhancing security to mitigate risks from FPGA firmware hacks and computer vision data poisoning attacks should be prioritized. Automotive-grade protection like firmware encryption, hardware root of trust, and improved anomaly detection for sensor inputs would limit vulnerabilities. This could prevent malicious actors from endangering passenger safety via compromised autonomous systems.

Combining FPGA flexibility and computer vision intelligence promises to usher in the next era of safe, efficient, and convenient self-driving transport[2]. But engineers must first surmount critical barriers in computational efficiency, corner case handling, functional safety, V2X integration, and cybersecurity before these innovations migrate successfully from laboratories to showrooms. Additional public-private partnerships, government initiatives like the Infrastructure Investment and Jobs Act which devotes billions towards intelligent transportation systems, and cross-industry standards bodies like the Autonomous Vehicle Computing Consortium will help accelerate progress on delivering robust FPGA and computer vision-enabled autonomous driving capabilities to the masses. Both hardware and software innovations in computing, communications and security will be integral for realization. While challenges persist, concentrated efforts along these vectors would solidify the integral role innovative FPGA and computer vision approaches will play in the autonomous vehicles of tomorrow.

REFERENCES

[1] S. Liu, J. Tang, Z. Zhang and J. -L. Gaudiot, "Computer Architectures for Autonomous Driving," in Computer, vol. 50, no. 8, pp. 18-25, 2017, doi: 10.1109/MC.2017.3001256.

[2] D. Castells-Rufas et al., "A Survey of FPGA-Based Vision Systems for Autonomous Cars," in IEEE Access, vol. 10, pp. 132525-132563, 2022, doi: 10.1109/ACCESS.2022.3230282.

[3] E. Jones, K. Pepper, A. Li, S. Li, Y. Zhang and D. Bailey, "Autonomous Driving Developed with an FPGA Design," 2019 International Conference on Field-Programmable Technology (ICFPT), Tianjin, China, 2019, pp. 431-434, doi: 10.1109/ICFPT47387.2019.00085.

[4] J. Borrego-Carazo, D. Castells-Rufas, E. Biempica, and J. Carrabina, ‘‘Resource-constrained machine learning for ADAS: A systematic review,’’ IEEE Access, vol. 8, pp. 40573–40598, 2020.

[5] D. Castells-Rufas, A. Saa-Garriga, and J. Carrabina, ‘‘Energy efficiency of many-soft-core processors,’’ in Proc. Int. Workshop High Perform. Energy Efficient Embedded Syst., 2016, pp. 1–8.

[6] K. Wei, K. Honda, and H. Amano, ‘‘FPGA design for autonomous vehi-cle driving using binarized neural networks,’’ in Proc. Int. Conf. Field-Programmable Technol. (FPT), Dec. 2018, pp. 428–431.

[7] H. Simmonds, N. Carlisle, X. Li, F. Mu and D. Bailey, "Autonomous vehicle development using FPGA for image processing", 2019 International Conference on Field-ProgrammableTechnology (ICFPT), pp. 449-452, 2019, [online] Available: https://doi.org/10.1109/icfpt47387.2019.00090.

[8] G. M, S. R, A. Kalaiselvi, U. M. V and M. A, "Implementation of Lane detection in Autonomous vehicle using FPGA," 2022 International Conference on Emerging Trends in Engineering and Medical Sciences (ICETEMS), Nagpur, India, 2022, pp. 141-147, doi: 10.1109/ICETEMS56252.2022.10093650.

[9] E. E. Oma, J. Zhang and Z. Lv, "FPGA Based Traffic Sign Detection Using Support Vector Machine and Hybrid Filters," 2022 10th International Conference on Intelligent Computing and Wireless Optical Communications (ICWOC), Chongqing, China, 2022, pp. 45-49, doi: 10.1109/ICWOC55996.2022.9809904.

[10] C. Gamez Serna and Y. Ruichek, "Traffic signs detection and classification for European urban environments", IEEE Trans. Intell. Transp. Syst., vol. 21, no. 10, pp. 4388-4399, Oct. 2020

[11] K. Harada, K. Kanazawa and M. Yasunaga, "FPGA-Based Object Detection for Autonomous Driving System," 2019 International Conference on Field-Programmable Technology (ICFPT), Tianjin, China, 2019, pp. 465-468, doi: 10.1109/ICFPT47387.2019.00094.

[12] T. Li, Y. Ma, H. Shen and T. Endoh, "FPGA Implementation of Real-Time Pedestrian Detection Using Normalization-Based Validation of Adaptive Features Clustering," in IEEE Transactions on Vehicular Technology, vol. 69, no. 9, pp. 9330-9341, Sept. 2020, doi: 10.1109/TVT.2020.2976958.

[13] C. Hao et al., "A Hybrid GPU + FPGA System Design for Autonomous Driving Cars," 2019 IEEE International Workshop on Signal Processing Systems (SiPS), Nanjing, China, 2019, pp. 121-126, doi: 10.1109/SiPS47522.2019.9020540.

[14] Gautam, S., & Kumar, A. (2023). Image-Based Automatic Traffic Lights Detection System for Autonomous Cars: A Review. Multimedia Tools Appl., 82(17), 26135–26182.