

Hardware-Efficient Comparative Framework for Relative and Absolute Depth Estimation on Embedded Vision Platforms

Divya Kiran¹, Ugra Mohan Roy²

Department of Electronics and Communication Engineering, M S Ramaiah University of Applied Sciences, Bangalore, India.

E-mail: ¹divyakiran.ec.et@msruas.ac.in, ²mohanroy.ec.et@msruas.ac.in

Orcid ID: ¹0000-0001-5925-5954, ²0000-0001-8596-6450

Abstract

One of the most crucial tasks of computer vision systems in autonomous navigation, robotics, obstacle avoidance, and real-time situational awareness is depth estimation. However, using depth estimation algorithms on embedded platforms can be limited by computational complexity, hardware resource usage, latency, and power consumption. This study provides a comparative analysis of relative and absolute depth estimation techniques from the perspective of hardware efficiency for embedded vision systems. The proposed relative depth estimation framework is based on a lightweight run-based connected component labeling (CCL) architecture to identify regions (objects) and infer ordinal depth relationships using area and centroid features. In contrast, the absolute depth estimation pipeline combines object localization with disparity-based metric depth estimation using camera geometry. Both approaches are tested on KITTI, NYU V2 and self-generated datasets, under FPGA and System-on-Chip (SoC) deployment environments. The experimental results show that the relative depth estimation framework consumes significantly fewer computational resources, with 3460 LUTs, 4 BRAMs, a power consumption of 0.78 W, and a processing latency of 0.30 ms/frame. The absolute depth estimation pipeline, on the other hand, consumes more hardware resources, such as approximately 18k LUTs, 28 BRAMs, 1.8 W power, and 3.80 ms/frame latency from the disparity estimation and CNN-assisted processing pipeline. The comparative analysis shows that relative depth estimation offers a better latency–power–hardware trade-off for resource constrained embedded platforms, while absolute depth estimation has better metric depth accuracy at the cost of higher computational complexity. The proposed framework demonstrates how lightweight relative depth inference can be used in real-time embedded perception applications.

Keywords: Monocular Depth Estimation, Relative Depth Estimation, Absolute Depth Estimation, Hardware Accelerator, Computer Vision.

1. Introduction

Depth estimation is key for autonomous navigation systems, robotics, obstacle avoidance, and intelligent transportation systems in the field of computer vision. It allows machines to recognize the three-dimensional shape of a scene by reading two-dimensional

image information, helping them understand space, locate objects and be aware of their surroundings [1]. For autonomous driving, obstacle avoidance, robotics, and intelligent transportation systems, depth estimation is an important component of the perception function. It is deployed in tasks like obstacle detection, scene interpretation, and distance estimation, and it directly affects navigation and safety decisions [2],[3].

The methods of depth estimation developed to date can be classified as either relative depth estimation or absolute depth estimation. Relative depth estimation is a way of finding the ordinal relationship between objects in a scene, i.e. whether an object is closer or further compared to another object without computing their actual metric distances. These solutions are computationally inexpensive and suitable for embedded vision systems that require fast scene understanding. In contrast, absolute depth estimation involves physical distance estimation based on disparity estimation, stereo geometry, or deep learning-based depth inference models [4]. While absolute depth estimation provides metric distance information, it typically requires more computations, memory bandwidth, and power to densely compute disparities and process them using CNN-based methods. In recent years, depth estimation systems have improved in precision using monocular and stereo cameras. Convolutional neural networks (CNNs) have been successful in depth predictions for autonomous perception tasks, and architectures incorporating CNNs and disparity learning have achieved significant improvements in accuracy. However, most current methods prioritize depth accuracy and overlook hardware efficiency, embedded deployment feasibility, and real-time implementation constraints. To be deployed at the edge in lightweight resource-constrained platforms like FPGAs or Systems-on-Chip (SoCs), vision systems must operate with minimal latency, memory and power requirements. Several recent studies have focused on improving cross-dataset generalization and robustness of monocular depth estimation models [5]. More recently, geometric scene modeling approaches have incorporated planar constraints to improve depth consistency and structural understanding [6].

In this work, relative and absolute depth estimation approaches are compared in terms of hardware efficiency for embedded vision applications. The relative depth estimation framework is based on a lightweight run based connected component labeling (CCL) architecture that allows for the extraction of the foreground objects and the inference of ordinal depth relationships based on object area and centroid information. The absolute depth estimation framework is based on disparity-based metric depth estimation and object localization techniques. All these approaches are tested against FPGA and SoC implementation parameters such as latency, power, LUTs, BRAMs, and processing speed.

The main contributions of this work are summarized as follows:

1. An embedded implementation framework for relative and absolute depth estimation techniques.
2. A lightweight, run-based Connected Component Labeling architecture for ordinal depth inference.
3. Evaluation based on hardware metrics such as latency, power, LUT, and BRAM usage.
4. Feasibility study of deploying FPGAs and SoCs in real-time embedded vision systems.

The experimental results show that relative depth estimation is more efficient in terms of latency, power and hardware consumption in resource constrained embedded platforms, and absolute depth estimation is beneficial in terms of metric depth at the cost of higher computational complexity.

2. Literature Survey on Existing Relative and Absolute Depth Estimation

Due to its applications in autonomous driving, robotics, augmented reality, obstacle avoidance, intelligent transportation systems, and human–robot interaction, depth estimation is one of the most important perception tasks in computer vision. Based on the output representation, current methods can be divided into two major classes: relative depth estimation (ordinal) and absolute depth estimation (metric). Relative depth estimation infers the order of objects without measuring the physical distance between them, while absolute depth estimation measures these distances in meters via stereo geometry or deep neural networks. Although estimation accuracy has been greatly enhanced by deep learning developments, the use of a high number of computational resources, high demand for hardware utilization, memory bandwidth, and energy consumption are still major challenges for an embedded vision platform. Many existing studies on monocular depth estimation were supervised by CNN-based methods. First, Eigen et al. showed that multi-scale CNNs can successfully estimate scene-depth from a single RGB image while simultaneously minimizing the relative depth errors, which are scale-invariant, using a new loss function [1]. Liu et al. also proposed a deep convolutional neural network and conditional random field model to maintain the boundaries and local structure of objects during dense depth prediction [7]. Such early works proved the viability of monocular depth estimation based on deep learning, but they demanded huge numbers of well annotated depth maps. Since ground-truth depth information was not essential, researchers came up with self-supervised learning approaches using stereo image reconstruction and view synthesis. Godard et al. presented Monodepth, a novel monocular depth learning approach that does not need dense depth labels by using left-right image consistency [4]. To address the drawbacks of this method, a new approach was developed, Monodepth2, which added photometric consistency, auto masking and minimum reprojection loss, thus achieving much better prediction accuracy and robustness in a variety of outdoor settings [8]. Later research further improved monocular depth estimation by improving camera pose estimation, uncertainty modelling, domain adaptation, and cross-dataset generalization [5] and [9]. Several researchers reported that the precise estimation of depth in embedded perception is not always a necessary requirement for functions such as obstacle prioritization or collision avoidance. Thus, the estimation of relative ordinal depth has gained in importance. Zoran et al. solved depth estimation as a pairwise ranking problem and showed that ordinal relationships provide reliable supervisory information if there is no metric ground truth [10]. To handle depth discontinuity and structural consistency, Fu et al. introduced the concept of the Deep Ordinal Regression Network (DORN) and changed the continuous depth estimation problem into an ordinal classification problem in their research [11]. Recently, ordinal constraints and ranking-based supervision were also added to self-supervised methods to enhance geometric consistency and remove scale ambiguity [12].

In recent times, architecture has matured to transformer-based models that can learn long-range contextual dependencies. Compared with traditional CNN-based models, Vision Transformers have shown their strong dense prediction capability by modeling global

spatial relationships better [13]. To extend monocular depth estimation to the next level of performance, Dense Prediction Transformer (DPT) [14] was developed, which uses a transformer network for feature extraction and a convolutional network for refinement, with impressive results on both the KITTI and NYU Depth V2 datasets. More recently, Depth Anything used large-scale unlabelled images and foundation-model pre-training to perform remarkable cross-domain generalization without the need for many labelled images [15]. These methods significantly enhance prediction accuracy but require considerable computation, which makes direct implementation on FPGAs embedded in other applications difficult. It is thus an active research area to achieve hardware efficient depth estimation. Wofk et al. used MobileNet encoders in combination with simple decoder designs to obtain real-time monocular depth estimation on embedded processors with a low computational load [16]. To reduce computational burdens, Lite-Mono adopted light-weight convolutional blocks and efficient feature aggregation while still achieving competitive accuracy [17]. Nevertheless, CNN inference is still quite costly, compared to the classical image-processing method, in terms of computational resources. Streaming architecture continues to be a desirable option for embedded vision systems based on FPGAs due to their deterministic latency, high parallelism, and low power consumption. One of the basic operations for object extraction and segmentation is called Connected Component Labelling (CCL). He et al. then introduced Connected Component Labelling based on the notion of run instead of pixel processing, thus reducing the number of memory accesses and the amount of computation required [18]. The run-based CCL has been shown to be efficient in terms of both its execution speed and hardware utilization and thus is well suited for embedded vision applications such as object detection, tracking, and region analysis [19, 20].

Numerous works on both relative and absolute depth estimation have been done, but most of the currently published literature is particularly concerned with prediction accuracy, relying on more complex neural network architectures. There are comparatively few investigations that look at such approaches from the perspectives of embedded hardware implementation such as FPGA resource utilization, BRAM consumption, DSP utilization, processing latency, memory bandwidth, computational complexity, and power consumption. Also, relatively lightweight depth estimation frameworks for relative depth estimation have not been compared directly with CNN-based frameworks for absolute depth estimation in the same embedded deployment scenarios. Inspired by these observations, the present work introduces a unified hardware-oriented comparative framework for embedded vision systems. The proposed framework combines a lightweight run-based Connected Component Labeling (CCL) architecture for ordinal depth inference with a CNN based Disparity Estimation (DE) pipeline for obtaining metric depth. In contrast to previous studies which tend to focus on prediction accuracy, this work comparatively analyzes latency, computational complexity, FPGA resource usage, BRAM usage, DSP usage, power consumption and deployability on FPGA and heterogeneous System-on-Chip platforms. The proposed framework thus offers practical design guidelines to determine appropriate depth estimation algorithms for embedded vision applications with limited resources.

3. Implemented Depth Estimation Frameworks

Depth estimation is a critical component of embedded vision systems used in autonomous navigation, robotics, obstacle avoidance, and intelligent surveillance applications.

Depending on the application requirements, depth estimation methods are generally categorized into relative depth estimation and absolute depth estimation approaches. Relative depth estimation determines ordinal spatial relationships among objects without computing exact physical distances, whereas absolute depth estimation estimates metric distances using disparity information and camera geometry. Relative depth estimation methods are computationally lightweight and are therefore suitable for FPGA-based edge devices and low-power embedded systems. In contrast, absolute depth estimation techniques provide improved spatial understanding and metric accuracy but require significantly higher computational complexity, memory bandwidth, and processing resources due to CNN-assisted disparity estimation and dense feature processing.

The proposed work presents a comparative embedded implementation framework analyzing the trade-offs between relative and absolute depth estimation approaches with respect to computational complexity, hardware utilization, memory requirement, processing latency, power consumption, and deployment feasibility on embedded systems. The framework specifically focuses on FPGA and System-on-Chip (SoC) implementation efficiency for real-time vision systems.

Table 1. Comparison between relative and absolute depth estimation approaches for embedded vision systems

Sl. No.	Relative Depth Estimation	Absolute Depth Estimation
1	Ordinal depth relationship	Metric distance estimation
2	Low computational complexity	High computational complexity
3	FPGA Compatible	Hardware Exhaustive
4	Low memory usage	High memory bandwidth
5	Low power consumption	Higher power consumption
6	Real-time edge deployment	Accuracy-oriented deployment

3.1 Top-level Framework

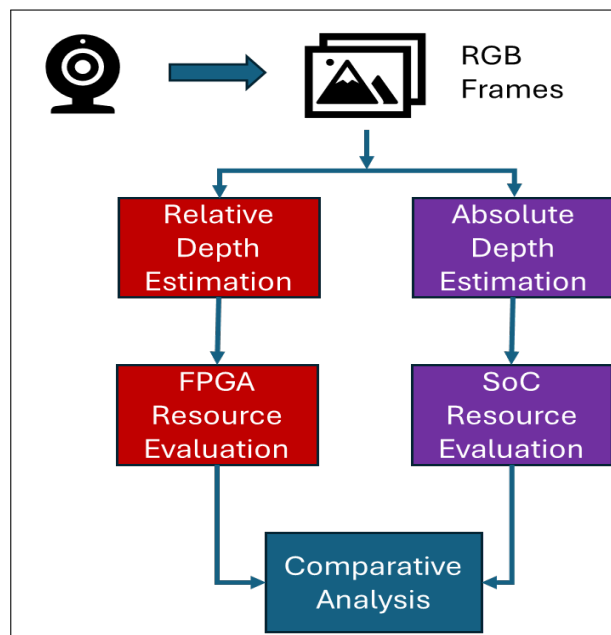


Figure 1. Overall comparative framework for embedded relative and absolute depth estimation

The proposed comparative framework, as shown in Figure 1 considers both relative and absolute depth estimation pipelines and implements them in an embedded environment. Lightweight ordinal depth inference and disparity-based metric depth estimation pipelines are used to process the same input image stream independently. The hardware usage and the processing efficiency are then compared. This architecture allows for determining appropriate depth estimation methods for resource-limited embedded systems based on application specific latency, accuracy, and hardware requirements.

3.2 Relative Depth Estimation Framework

Relative depth estimation as illustrated in Figure 2 determines the ordinal spatial relationship among objects without calculating exact physical distances from the camera. The proposed framework employs a lightweight run-based Connected Component Labeling (CCL) architecture for extracting foreground regions and inferring relative object depth using area and centroid information [21].



Figure 2. Pictorial representation of Relative Depth estimation

Relative depth estimation works to estimate ordinal depth relationships among the objects present in a scene without calculating their actual physical distance to the camera. Rather than measuring distance in a metric unit, it establishes the proximity of one object to another, determining whether it is nearer or more distant. This paradigm is especially beneficial for embedded vision and real-time systems with low computational resources and power budgets. Relative depth can be estimated with the help of the key cues listed below, which are similar to those of human visual perception and allow depth argumentation through the prism of monocular vision:

- Obvious size of the object (greater size is supposed to be closer)
- X- Z position – in the image plane.
- Occlusion and overlap
- Frame to frame centroid displacement.

The framework is designed specifically for FPGA-oriented streaming implementation with reduced memory accesses and low computational complexity. As depicted in Figure 3 the relative depth estimation pipeline consists of the following stages:

1. Image pre-processing and segmentation
2. Run formation
3. Connected Component Labeling
4. Area and centroid computation
5. Relative depth ranking

Foreground objects occupying larger image regions and positioned lower in the image plane are generally interpreted as being closer to the camera.

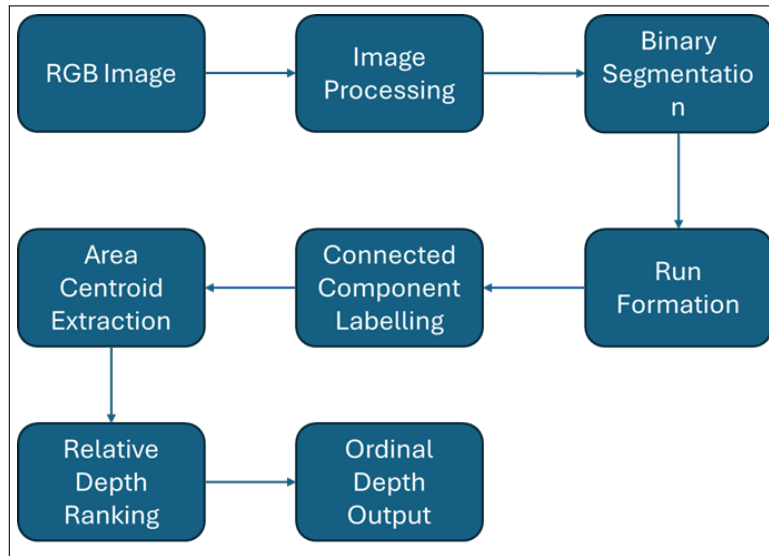


Figure 3. Pipeline of the proposed run-based relative depth estimation framework

3.2.1 Image pre-processing and Binary segmentation

The RGB image is reduced in computational complexity and hardware implementation are reduced by first converting it to an 8-bit grayscale image, as shown in Figure 4. A 3×3 median filter is used to remove impulse noise while retaining object edges. For foreground extraction, an adaptive thresholding method based on Otsu's technique is employed. This method automatically calculates an optimum threshold to maximize the inter-class variance between foreground and background pixels. The automatically computed threshold value for the datasets used in this work (KITTI, NYU V2, and the self-generated dataset), ranged between 118 and 146 gray levels with an average for 8-bit images of $TH = 132$. These values were determined to be effective for achieving good foreground segmentation without the need for manual adjustment for different illumination conditions. A binary image is then created based on Eq. 1,

$$B(x, y) = \begin{cases} 1, & I(x, y) \geq T \\ 0, & I(x, y) < T \end{cases} \quad (1)$$

where $I(x, y)$ is the gray level intensity and T is the adaptive threshold computed using Otsu's algorithm. After thresholding, morphological opening with a 3×3 square structuring

element eliminates isolated noisy pixels and morphological closing with the same structuring element fills small holes in foreground objects. Regions with an area less than 25 pixels are removed to prevent the addition of "noise blobs" which may impact centroid calculation and object ranking. The resulting binary image is then passed to the run-generation module where all the contiguous foreground pixels are converted to runs prior to Connected Component Labeling (CCL).

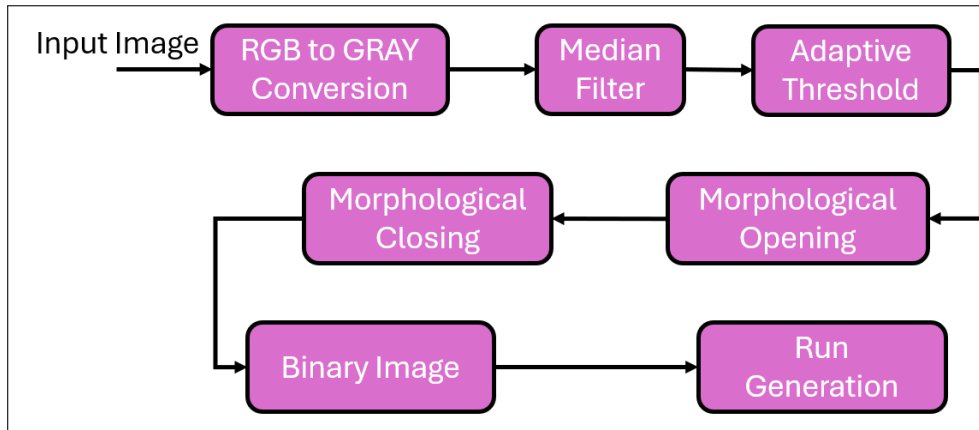


Figure 4. Pipeline of the proposed run-based relative depth estimation framework

3.2.2 Mathematical Model for Relative Depth Inference

Relative depth estimation is modeled as an ordinal ranking problem over detected objects.

Let the set of detected objects be:

$$\mathcal{O} = \{O_1, O_2, \dots, O_N\} \quad (2)$$

For each object O_i , the following features are extracted:

- Area: A_i
- Centroid: (C_{x_i}, C_{y_i})

Relative depth score D_i for object O_i is computed as a weighted function:

$$D_i = \alpha \cdot \frac{A_i}{\max(A)} + \beta \cdot \frac{C_{y_i}}{H} \quad (3)$$

where:

- H is image height,
- α, β are weighing coefficients ($\alpha + \beta = 1$).

Objects are ranked based on descending D_i , where a higher score indicates closer proximity. The weighting coefficients were empirically obtained by performing grid-search over the validation set. The values of α and β were changed from 0 to 1 with a step of 0.1 and at the same time $\alpha + \beta = 1$. The highest accuracy in the ordinal ranking was achieved for $\alpha = 0.6$ and $\beta = 0.4$.

3.2.3 Run-Based Connected Component Labeling

The equations 4-8 adopted have been validated by the previous work [21, 22]. The connectivity rules have been applied for region of interest detection and centroid calculation. The previous work explains the validation of these equations and suggests it as a hardware efficient approach for the region of interest detection with their coordinates.

Run Formation and Representation

A run is defined as a contiguous sequence of foreground pixels in a single row. For a given image row i , a run $R(i, j)$ is represented as:

$$R(i, j) = RS(i, j) \cup RE(i, j) \quad (4)$$

where:

- $RS(i, j)$ denotes the start column of the run
- $RE(i, j)$ denotes the end column of the run.

Inter-Row Connectivity Analysis

Connectivity between runs in consecutive rows is established by comparing their start and end positions. Let:

- R_{CR} be the run in the current row,
 - R_{PR} be the run in the previous row.
1. **Start–Start Difference (Diff_{SS})** – It measures the horizontal displacement between the start positions of the runs in the current and previous rows. A positive value indicates that the current run starts to the right of the previous run. A negative value indicates that it starts to the left.

$$\text{Diff}_{SS} = RS_{CR} - RS_{PR} \quad (5)$$

2. **End–End Difference (Diff_{EE})** – It represents the displacement between the end positions of the two runs. A positive value implies the current run ends to the right of the previous run. A negative value implies it ends to the left.

$$\text{Diff}_{EE} = RE_{CR} - RE_{PR} \quad (6)$$

3. **Start–End Difference (Diff_{SE})** – It checks whether the start of the current run overlaps or lies beyond the end of the previous run. If it is positive, the current run starts after the previous run ends (no overlap on the left side). If negative, the current run overlaps or lies within the previous run.

$$\text{Diff}_{SE} = RS_{CR} - RE_{PR} \quad (7)$$

4. **End–Start Difference (Diff_{ES})** – It checks whether the end of the current run overlaps or extends beyond the start of the previous run. If it is negative, the current run ends before the previous run begins (no overlap). If positive, there is an overlap or adjacency.

$$\text{Diff}_{ES} = RE_{CR} - RS_{PR} \quad (8)$$

- RS_{CR} = Start column of a run in the current row
- RE_{CR} = End column of a run in the current row
- RS_{PR} = Start column of a run in the previous row
- RE_{PR} = End column of a run in the previous row

5. Connectivity Rule Conditions

- If $Diff_{SS}$, $Diff_{EE}$, $Diff_{SE}$, and $Diff_{ES}$ are positive respective rows are not connected.
- If $Diff_{SS}$, $Diff_{EE}$, $Diff_{SE}$, and $Diff_{ES}$ are negative respective rows are not connected.
- If $Diff_{SS}$ is positive and $Diff_{EE}$ is negative or vice versa rows are connected.
- If $Diff_{SE}$ is positive and $Diff_{ES}$ is negative or vice versa rows are connected.

0	0	1	1	1	0
0	1	1	1	0	2
0	0	0	0	2	2
3	3	3	0	2	2
3	3	0	0	2	2
3	0	0	0	0	0

Figure 5. Example of labelling connected pixels

Figure 5 shows an example of pixel labelling using the implemented connectivity rules. Once connectivity is checked, connected “1s” are assigned the same label, while disconnected “1s” are assigned different labels. Finally, different objects with distinct labels are identified. Figure 5 illustrates the labelling of connected pixels. Section 4.2 describes the centroid calculation and also validates that this proposed approach provides accurate object detection.

3.2.4 Computational Complexity Analysis

The proposed run-based Connected Component Labeling (CCL) algorithm aims to lower the complexity by processing runs of contiguous foreground pixels rather than individual pixels. Unlike traditional pixel-based CCL algorithms which check each foreground pixel and its neighbouring pixels, the proposed algorithm represents each connected foreground region by its starting and ending coordinates, thus decreasing the number of memory accesses and neighbourhood comparisons. In conventional CCL, the complexity of the computational complexity is $O(W \times H)$ because every pixel is scanned and labeled. The proposed run-based CCL is based on the number of foreground runs, R , which has a computational complexity of $O(R)$. A foreground run is composed of several contiguous foreground pixels, so the number of runs is significantly smaller than the number of pixels ($R \ll W \times H$), thus offering lower computational overhead and memory demands. In this work the FPGA implementation carries out three main operations for every row of the 100×100 test image; While the binary image is being processed, execute the following operations: connectivity comparison to the previous line and assigning labels to centroids using a centroid accumulation approach. The average

processing requirement per measured image row was about 300 clock cycles with a total of about 30,000 clock cycles for the entire 100×100 image. The execution time at an operating frequency of 100 MHz at:

$$\frac{30,000}{100 \times 10^6} = 300 \mu s = 0.30 \text{ ms} \quad (9)$$

This corresponds to the processing latency stated in Table 6. Most of the reduction in computational complexity comes from processing foreground runs instead of processing individual pixels, which results in a reduction in the number of neighborhood comparisons and label-equivalence operations. The proposed architecture can be implemented efficiently in streaming FPGA with compact memory constraints. A comparison of CCL approaches based on pixel and run is presented in Table 2.

Table 2. Comparison of conventional and proposed CCL approaches

Parameter	Conventional Pixel-Based CCL	Proposed Run-Based CCL
Processing unit	Individual pixel	Foreground run
Computational complexity	$(O(W \times H))$	$(O(R))$
Image size	100×100	100×100
Measured clock cycles	Higher (pixel dependent)	30,000 cycles
Average cycles per row	Pixel dependent	≈ 300 cycles
FPGA implementation	Multiple neighborhood accesses	Streaming architecture
Memory requirement	High	Low

3.2.5 Pseudo Code for Connected Pixel Labelling

-
1. Initialize label_counter $\leftarrow 1$
 2. For each row i from 1 to n :
 - a. Detect runs R_i
 - b. Assign temporary labels to runs
 - c. Compare runs with row $i-1$
 - d. If connected:
 - Merge labels using equivalence class
 3. Resolve equivalence classes using minimum label
 4. Relabel image using resolved labels
 5. Return labeled image L
-

3.2.6 Implemented Connected Labelling Approach

Connected Component Labelling (CCL) is a basic image processing procedure that is utilized to recognize and label connected foregrounds. Classical pixel-based CCL has a computational complexity which is very high as it requires repetitive neighborhood tests and equivalence resolution.

The connected component labeling run-based algorithm was meant to minimize:

- Computational complexity,
- Memory usage
- Number of equivalence conflicts.

The novel approach of connected labelling has been used in this work thus making it

more effective to implement hardware [14]. Figure 6 shows the centroid computation flow diagram.

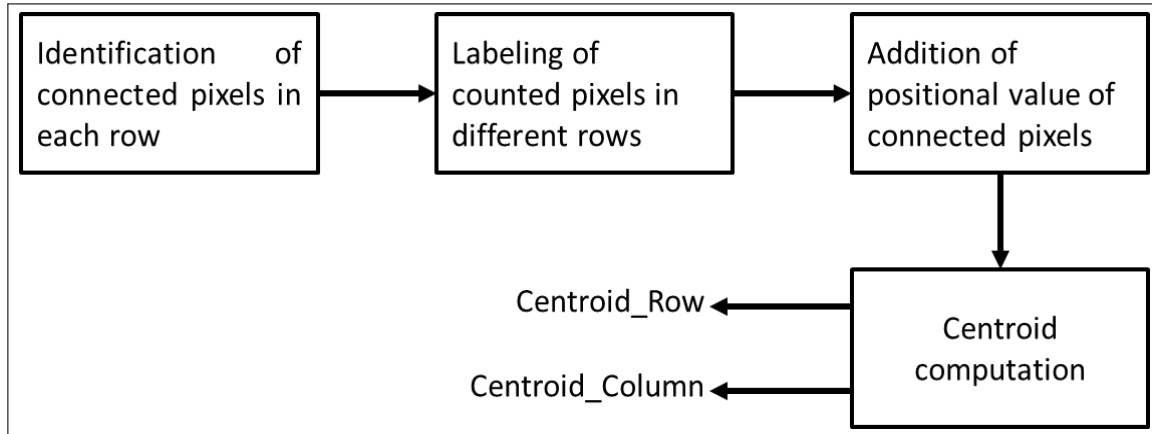


Figure 6. Centroid computation

The pre-processing step prepares the image for blob detection. All images are handled as matrices, with each position representing a pixel. Data compression is achieved using run-length encoding, and the coded data is received in the form of start and end points in the rows. The CCL technique is then applied for labeling, and moments are used to determine the area and centroid of the labeled regions.

3.2.7 Area and Centroid Computation

For each connected component, the object area is computed as:

$$\mu_{0,0} = \sum_{x=0}^w \sum_{y=0}^w f(x,y) \quad (10)$$

Centroid Row

$$C_{x_i} = \frac{sum_x}{\mu_{0,0}} \quad (11)$$

Where, $sum_x = \sum \sum x f(x,y)$

Centroid Column

$$C_{y_i} = \frac{sum_y}{\mu_{0,0}} \quad (12)$$

Where, $sum_y = \sum \sum y f(x,y)$

Figure 7 is the example of centroid computation as per the equations. Repetitive summation has been adopted to find centroid. For “area” moment $\mu_{0,0}$ has been calculated as shown in equation 10 and for “centroid” first $\mu_{0,1}$ that is sum_x and $1,0$ that is sum_y are calculated and divided by $\mu_{0,0}$ illustrated in equations 11 and 12. The same approach has been adopted for real time implementation on hardware.

0	0	1	C	1	0
0	1	1	1	0	2
0	0	0	0	2	2
C	3	3	0	2	C
3	3	0	0	2	2
3	0	0	0	0	0

Figure 7. Example of centroid computation

3.3 Absolute Depth Estimation Framework



Figure 8. Pictorial representation of absolute depth estimation

Absolute depth estimation computes the physical distance between objects and the imaging sensor using disparity-based metric depth computation, as shown in Figure 8. It generates metric depth values, in most cases in meters, unlike relative depth estimation. The method is necessary for applications that need accurate distance measurements, such as autonomous driving and collision avoidance systems. Absolute depth estimation is based on the principles of camera geometry and photogrammetry. The disparity-based depth estimation method is the most prevalent one, in which depth is proportional to the difference between matching pixels in stereo or monocular depth estimation systems. Recent systems incorporate deep learning models to predict disparity using a single image and then generate depth values using geometric computation.

The implemented framework as illustrated in Figure 9 integrates:

1. Object localization
2. Disparity map generation

3. Centroid mapping

4. Metric depth estimation

The framework employs CNN-assisted disparity estimation together with geometric depth computation using camera focal length and stereo baseline parameters. Object localization is performed using a CNN-based detection framework like Mask R-CNN [23]. Recent self-supervised monocular depth estimation methods have further improved depth consistency through enhanced pose estimation strategies [24].

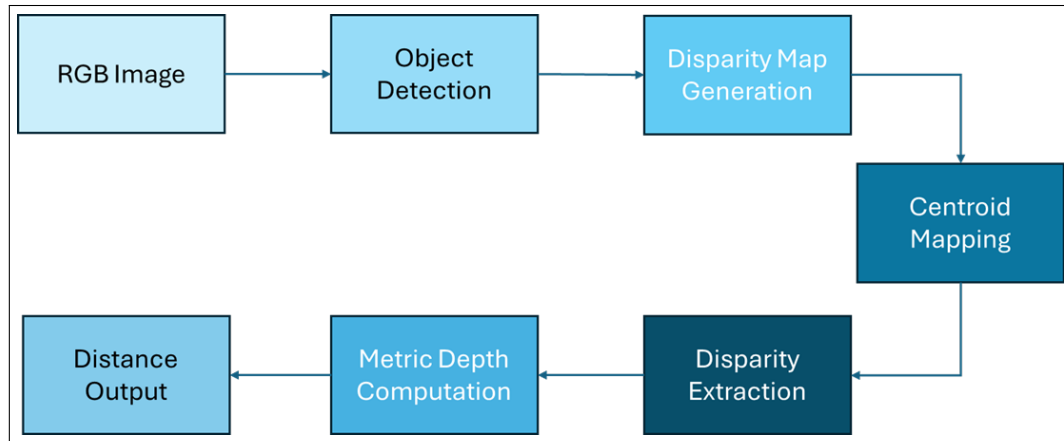


Figure 9. Pipeline of the implemented absolute depth estimation framework

3.3.1 CNN-Based Disparity Estimation Network

Our absolute depth estimation framework uses a pretrained Monodepth2 convolutional neural network that estimates dense depth from a single RGB image. Monodepth2 was chosen because it offers a good compromise among depth estimation accuracy, computational complexity and portability to embedded vision platforms. The network comprises an encoder (ResNet-18) and a lightweight decoder that includes successive up-convolution layers with skip connections. The main difference between the two is that the former (encoder) extracts an image feature hierarchy, while the latter (decoder) outputs a dense disparity map at the original image resolution. The network was tested on both KITTI and NYU Depth V2 datasets. Different image resolutions and scene characteristics exist for these datasets, and the images were resized to maintain the aspect ratio without loss of information before they could be fed to the network. The outdoor images of KITTI were resized to 640×192 pixels and the indoor images of NYU Depth V2 were resized to 640×480 pixels, which is the typical resolution for monocular depth estimation. The network parameters were initialized from pre-trained publicly available pre-trained weights from Monodepth2 and not retrained in this work. Therefore, the emphasis of the present study is on embedded implementation, disparity to depth conversion, localization of the objects from the centroid, and evaluation of the hardware performance, and not on the development of a new CNN architecture. The disparity map obtained is used to calculate the mean disparity in the identified object region. The disparity value of the centroid neighbourhood is then used to replace the disparity value in the stereo camera model to estimate the metric depth of each object detected. Table 3 and Table 4 show the architecture of the CNN and the configuration of the datasets.

Table 3. CNN architecture used for absolute depth estimation

Parameter	Value
CNN Model	Monodepth2
Encoder	ResNet-18
Decoder	Multi-scale up-convolution decoder
Input (KITTI)	640 × 192 RGB
Input (NYU V2)	640 × 480 RGB
Output	Dense disparity map
Backbone Parameters	≈11.7 million
Framework	PyTorch
Optimizer (original training)	Adam
Initial Learning Rate	1 × 10 ⁻⁴
Batch Size	12
Training Epochs	20
Loss Function	Photometric Loss + SSIM + Edge-aware Smoothness
Inference	Pre-trained network
Deployment	ARM Cortex + Zynq 7020

Table 4. Dataset configuration

Dataset	Training Images	Validation Images	Test Images	Resolution
KITTI Eigen Split	39,810	4,424	697	1242 × 375
NYU Depth V2	24,231	654	654	640 × 480
Self-Created Dataset	300	40	80	1024 × 1024

The disparity estimation accuracy was measured based on the standard metrics used by Monodepth2 on the KITTI Eigen split and NYU Depth V2 datasets. Generated disparity maps were then checked visually and used in centroid based metric depth estimation. The published pre-trained network was used without changes as the goal of this work was to focus on hardware implementation and comparative embedded deployment.

3.3.2 Disparity-to-Depth Mathematical Model

Depth at pixel (x, y) is computed as:

$$Z(x, y) = \frac{f \cdot B}{d(x, y)} \quad (13)$$

For object O_i , depth is estimated at the centroid location:

$$Z_i = \frac{f \cdot B}{d_i} \quad (14)$$

where d_i is the average disparity within a local neighborhood Ω_i :

$$d_i = \frac{1}{|\Omega_i|} \sum_{(x, y) \in \Omega_i} d(x, y) \quad (15)$$

- $d(x, y)$ be disparity value at pixel (x, y) ,

- f be camera focal length (pixels),
- B be baseline distance (meters).

3.3.3 Pseudo Code for Absolute depth estimation

1. Detect objects using object detection model
2. Generate disparity map using depth estimation CNN
3. For each detected object k :
 - a. Compute bounding box centroid (C_{xk}, C_{yk})
 - b. Extract disparity neighborhood Ω_k
 - c. Compute average disparity d_k
 - d. Compute depth $Z_k = (f \times B) / d_k$
4. Return depth values for all objects

This proposed approach for absolute depth estimation is computationally intensive and has been deployed on System-on Chip (SoC) architecture combining ARM processors and FPGA fabric to evaluate the performance [25].

3.4 FPGA-Oriented Embedded Hardware Architecture

The proposed framework is implemented using FPGA and heterogeneous SoC architectures to evaluate real-time embedded deployment feasibility.

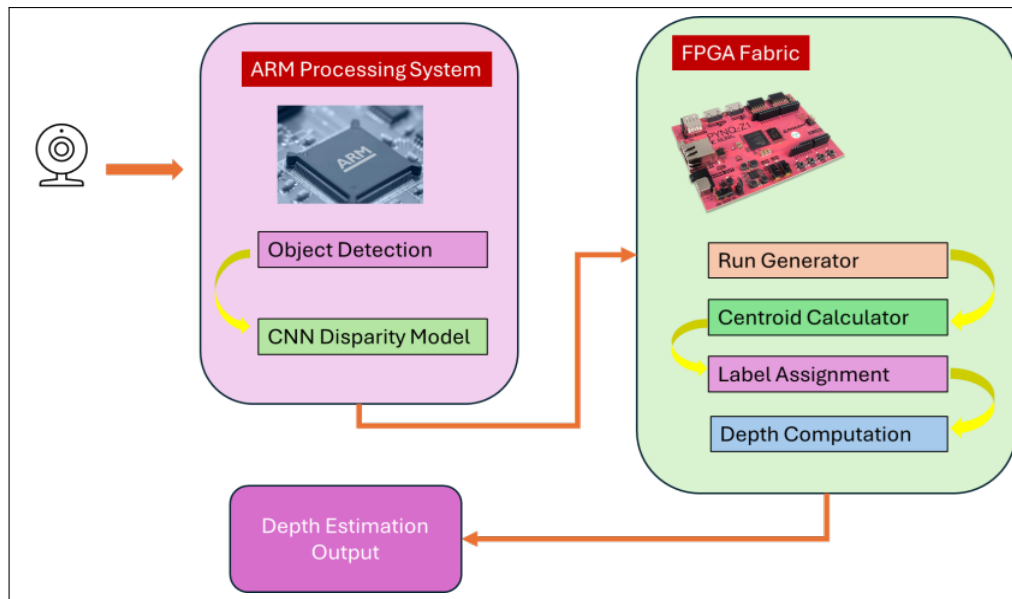


Figure 10. Embedded FPGA-SoC architecture for depth estimation

Figure 10 shows the relative depth estimation framework utilizes streaming FPGA architecture with lightweight image-processing modules including:

- run generator
- label assignment
- centroid calculator
- depth ranking unit

The absolute depth estimation framework employs ARM processors for CNN-assisted disparity estimation and FPGA acceleration for centroid mapping and depth computation.

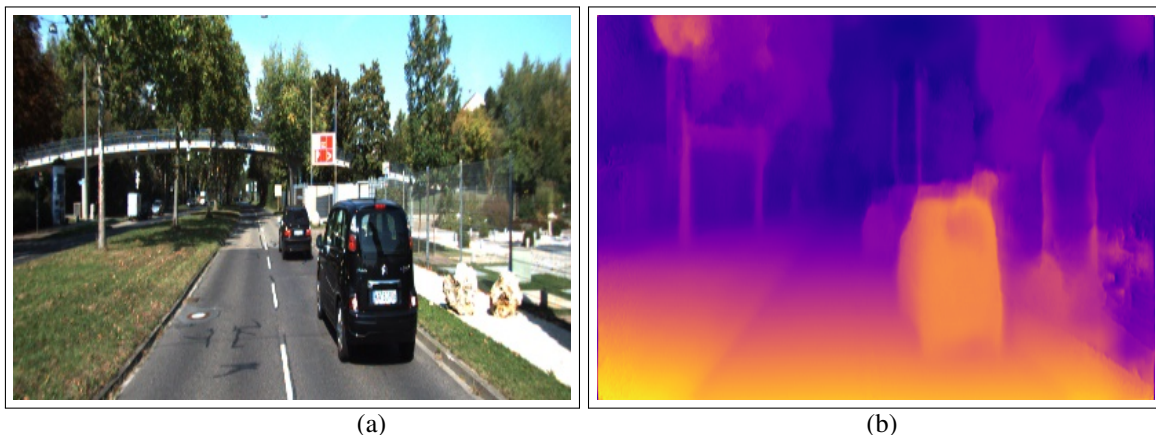
FPGA-based acceleration remains an effective approach for reducing latency and improving energy efficiency in deep-learning-enabled vision systems [26][27].

The proposed frameworks' hardware resource utilization and power consumption were synthesized and implemented using FPGA design tools and quantified after synthesis. The reported logic utilization is for the complete hardware architecture, run generation, Connected Component Labeling (CCL), centroid computation, control logic, and memory interfaces for the relative depth estimation framework. The 3460 LUTs, therefore, represent the logic utilization of the entire relative depth estimation pipeline, not just the centroid computing module. Likewise, the 4 BRAMs mentioned are also used for intermediate buffers, label memory and run information. A clock frequency of 100 MHz was used, and the measured dynamic power consumption of the complete FPGA implementation was 0.78W under nominal operating conditions. The absolute depth estimation framework uses the FPGA accelerator to perform disparity-to-depth conversion, centroid mapping, and control logic, while the CNN based disparity estimation was implemented on the ARM processing system of the heterogeneous SoC. Implementation took roughly 18,000 LUTs, 28 BRAMs and 1.80 W of measured dynamic power. These values are for the full implementation of the logic on the FPGA and thus reflect both the FPGA logic utilization and processing overhead of the embedded SoC platform.

4. Results and Discussions

4.1 Evaluation of Proposed Approaches

Recent benchmark studies emphasize the importance of evaluating depth estimation algorithms across diverse environments and operating conditions [28]. For the evaluation of the proposed approaches, the KITTI dataset, NYU V2 dataset, and a self-created dataset were used. These datasets include RGB and grayscale images with resolutions up to 1024x1024. The developed models were also run on an FPGA at a frequency of 100 MHz. Relative depth estimation is based on depth inference, while absolute depth estimation incorporates object detection, disparity estimation, and geometric depth computation. Centroid calculation was performed using connected component labeling, and absolute depth estimation was achieved using CNN-based inference. Relative and absolute depth estimation were tested for efficiency using four different metric performance studies, as discussed below. The obtained result snapshots are shown in Figure 11 and Figure 12.





(c)

Figure 11. Results obtained using Kitti dataset (a) test image (b) depth image (c) object of interest detection**Figure 12.** Results obtained using NYU V2 dataset

The experimental findings have shown that relative depth estimation pipelines use lesser clock cycles, less memory and less power compared to absolute depth estimation structures. Relative and absolute depth estimation techniques are summarized with a comparative analysis given in Table 5.

Table 5. Analysis of relative vs absolute depth estimation

Feature	Proposed Relative Depth Estimation	Absolute Depth Estimation
Primary Objective	Relative ordering of scene objects	Metric depth estimation
Output	Depth rank (Near/Far)	Absolute depth (m)
Ground Truth Requirement	Not required during inference	Required for training and evaluation
Typical Dataset	General monocular images	KITTI, NYU Depth V2
Input Resolution	100 × 100	KITTI: 640 × 192 NYU V2: 640 × 480
Computational Complexity	O(R) (Run-based CCL)	High (CNN inference)
Processing Cycles	30,000	380,000
Practical Throughput	180 FPS	22 FPS
FPGA Resources (LUTs)	3460	18,000
BRAM	4	28

DSP Blocks	0	32
Dynamic Power	0.78 W	1.80 W
Depth Accuracy Metric	Pairwise ranking accuracy	Abs Rel, RMSE, $\delta < 1.25$
Typical Application	Obstacle prioritization, robot navigation, collision avoidance	Autonomous driving, 3D reconstruction, AR/VR
Embedded FPGA Suitability	High	Moderate
Depth Output	Relative Rank	Metric Depth
Kernel Latency	0.30 ms	3.80 ms
Operating Frequency	100 MHz	100 MHz
Target FPGA	Zynq 7020	Zynq 7020 (ARM + FPGA)

To reflect the proposed relative depth- estimation framework's design objectives instead of a direct comparison of depth accuracy, these two frameworks are compared in Table 5. The proposed approach adopts a run-based Connected Component Labeling algorithm that has an operation complexity of $O(R)$ (where R is the number of runs), needs only 3460 LUTs, 4 BRAMs and 0.78 W of dynamic power while producing a practical throughput of around 180 FPS for a 100×100 image. The CNN-based approach, however, uses higher-resolution KITTI or NYU Depth V2 images, necessitating much more computational power (18,000 LUTs, 28 BRAMs, 32 DSPs) and power (1.80 W), to achieve an achievable throughput of ~ 22 FPS. Thus, the two approaches are suitable for different application needs: the proposed framework focuses on computational efficiency in embedded real-time systems, while CNN-based methods focus on the accuracy of the metrics for scene reconstruction and autonomous navigation. The CNN-based absolute depth estimation uses a heterogeneous Zynq 7020 to perform deep learning inferences and geometry-based depth computations while the lightweight run-based relative depth estimation operates exclusively on FPGA fabric. Thus, the reported hardware parameters are not a direct comparison between the various FPGA devices but reflect the cost of implementing the respective algorithm. Figure 11 shows the comparison of latency and power for relative and absolute depth estimation.

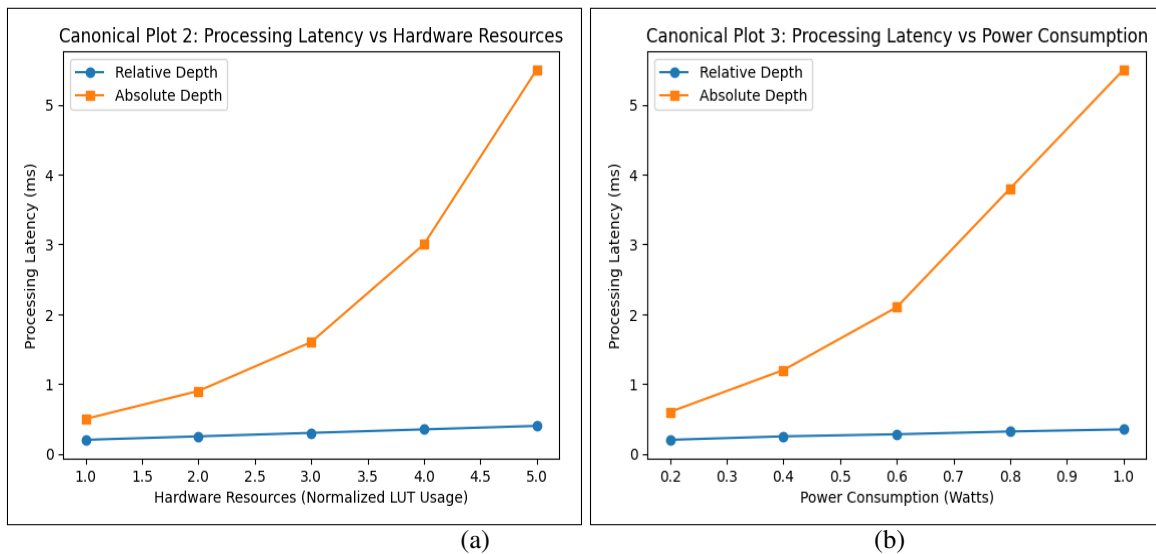


Figure 13. Comparison of latency of relative and absolute depth estimation vs (a) hardware resource utilization (b) power consumption

4.2 Centroid Calculation Result



Figure 14. Test image for centroid calculation

Table 6. Experimental results of test image (100 X 100)

Object number	Area	Centroid Row	Centroid Column
1	695	24	23
2	480	25	71
3	385	66	77
4	61	59	53
5	278	85	23
6	127	85	55

Figure 13 shows the test image, and Table 6 shows the calculated area and centroid. The algorithm consumes an average of 0.22ms to calculate the area and centroid of objects and for labeling. The computation module for the centroid used only 416 Slice LUTs, 56 I/O buffers, and shared 4 BRAMs in the full relative depth estimation system. The experimental results show that the connected component labeling (CCL) algorithm provides reliable object localization based on the area of components and the location of their centroid. When an object's spatial position changes, its centroid also changes, supporting the potential of using the centroid as a spatial feature for motion detection and visual navigation. The area of the detected component also provides an approximation of the object's apparent size. The experiments also demonstrate that CCL is useful in preserving positional information across changes in illumination and camera orientation. If the brightness or camera angle changes, the area the image occupies, and in some cases the detection of low-contrast parts of an object, may also change. However, the centroid coordinates of the correctly detected components will still represent the true image location. This means that this algorithm is more sensitive to the quality of segmentation than to the actual intensity of the image in determining the object's location.

When the image dimensions are increased by factors of 2 in the row direction and 7 in the column direction, the image area increases by approximately 14 $\times$. Similarly, the x and y coordinates of the centroid increase by about 2 and 7, respectively. Likewise, the x and y coordinates of the centroid cell also grow by a factor of 2 and 7, respectively. It has been experimentally observed that the scaling of these parameters confirms that the centroid calculation is consistent across different image resolutions.

4.3 Absolute Distance Calculation



Figure 15. Sample test images for distance calculation

Table 7. Distance Obtained for the Test image

Image	Objects in an Image	Actual Distance in m	Predicted distance in m	Error (%) = $\frac{ a-b }{a} \times 100\%$
Image1	car	9	9.31m	3.4%
Image2	car	6	7.3m	22%
Image 3	scooter	3.5	4.12	4%
	car	5.5	5.44	1.09%
Image 4	car	3.5	3.7	5.71%
Image 5	car	5	5.17	3.4%
Image 6	car	8.05	8.24	2.36%
Image 7	scooter	3.5	4.12	17.7%
	car	5.5	5.44	1.08%
Image 8	scooter	4	3.75	6.2%
	car	11	10.67	3%
Image 9	scooter	4.5	4.18	7.1%
	car	8	8.15	1.875%
Image 10	car	6.05	5.56	8.09%

$$\text{Mean Error (\%)} = \frac{1}{N} \sum_{i=1}^N \left(\frac{|A_i - P_i|}{A_i} \times 100 \right) \quad (16)$$

$$= 7.18\%$$

$$\text{Mean Absolute Error (MAE)} = \frac{1}{N} \sum_{i=1}^N |A_i - P_i| \quad (17)$$

$$= 0.362m$$

$$\text{Root Mean Square Error (RMSE)} = \sqrt{\frac{1}{N} \sum_{i=1}^N (A_i - P_i)^2} \quad (18)$$

$$= 0.478m$$

Figure 14 shows the test image sample, and Table 7 shows the estimated distance for 10 images out of 40 test images. The absolute depth estimation framework obtained a Mean Error of 7.18%, a Mean Absolute Error (MAE) of 0.36 m, and Root Mean Square Error (RMSE) of 0.48 m based on the evaluated test sample. The respective equations are mentioned in eqn 15-17. Results show that the metric depth estimation performance is reasonable for embedded perception. The absolute depth estimation framework can provide quantitative distance information with acceptable estimation accuracy while the relative depth framework requires less hardware complexity and power consumption.

Table 8. Min + Max Ranges

Method	Indoor Min	Indoor Max	Outdoor Min	Outdoor Max
Relative Depth (ordinal)	0.5 m	15 m	0.5 m	50 m
Absolute Depth (metric)	0.5 m	30 m	0.5 m	80 m

As shown in Table 7 and Table 8 the maximum disparity limits the estimates of absolute depth within a far range limited by $d_{min} \approx 1\text{pixel}$ and Near range limited by stable d_{min} and occlusion effects. Relative depth estimation range depends on object visibility, segmentation stability, camera FoV and resolution.

Table 9. Comparison with existing FPGA-based relative depth estimation

Existing Work	FPGA Platform	Function	Throughput/FPS	Hardware Resources
Benkrid et al. (2003) [29]	Xilinx FPGA	Connected Component Labeling	Real-time	FPGA implementation
Spagnolo et al. (2018) [30]	Zynq-7045	Connected Component Labeling + Feature Computation	220 Mpixels/s	<18,000 LUTs
Spagnolo et al. (2020) [31]	Zynq-7000	Parallel Connected Component Labeling	>30 fps (2048×2048)	3688 LUTs, 10 kB BRAM
Proposed Relative Depth Estimation	Zynq-7020	Segmentation + Run-based CCL + Centroid + Relative Ranking	180 fps (100×100)	3460 LUTs, 4 BRAM

Table 10. Comparison with existing FPGA-based absolute depth estimation

Existing Work	FPGA Platform	Network	Resolution	FPS	Power
Sada et al. (2020) [32]	Low-cost FPGA	DepthFCN	256×256	123.6	0.30 W
Hashimoto & Takamaeda-Yamazaki (2022) [33]	Xilinx ZCU104	DeepVideoMVS	Video	NR	NR
Li et al. (2026) [34]	ZCU104	RE-LFDE	Light-field images	NR	Resource-efficient
Proposed Absolute Depth Estimation	Zynq 7020	ResNet-18	640×192	22	1.80 W

Tables 9 and 10 compare the proposed framework quantitatively with some representative FPGA-based implementations for relative and absolute depth estimation. Existing FPGA-based relative depth estimation methods, which mainly deal with connected component labeling and feature extraction, are capable of high throughput (up to 220 Mpixels/s

or >30 fps) with up to 18,000 LUTs. In contrast, the proposed relative depth estimation framework only requires 3,460 LUTs, 4 BRAMs, and no DSP blocks and achieves 180 fps with 0.78 W when implemented on Zynq 7020. Existing FPGA implementations like DepthFCN can run at 123.6 fps with 0.30 W of power consumption by employing dedicated CNN accelerators. The implementation introduced herein on the heterogeneous Zynq 7020 can run the pre-trained Monodepth2 network at 22 fps with 18,000 LUTs, 28 BRAMs and 32 DSP blocks, consuming 1.80 W of power. From the comparisons, the proposed relative depth estimation framework has better hardware efficiency for resource constrained embedded systems while the absolute depth estimation framework shows the extra computational/hardware requirement to obtain metric depth estimation using deep learning.

4.4 Ablation and Statistical Validation

The results of the ablation study suggest that the area and centroid cues contribute to the relative depth ranking. Quantitative evaluation metrics have been applied as depicted in equation 19-21. Table 11 shows the results for the proposed relative depth estimation.

4.4.1 Quantitative Evaluation Metrics

1. Pairwise Ranking Accuracy gives the percentage of object pairs whose near/far relationship is correctly predicted

$$PRA = \frac{N_{correct}}{N_{pairs}} \times 100 \quad (19)$$

2. Kendall's Tau measures the agreement between two rankings based on pairwise ordering.

$$\tau = \frac{C - D}{\frac{1}{2}n(n-1)} \quad (20)$$

where:

- C= number of concordant pairs
 - D= number of discordant pairs
 - n = number of objects.
3. Spearman's Rank Correlation compares the rank positions assigned by the proposed method with those from the ground truth.

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \quad (21)$$

where d_i is the difference between predicted and ground-truth ranks.

Table 11. Ablation of the proposed relative depth estimation

Configuration	Area Weight (α)	Centroid Weight (β)	Pairwise Ranking Accuracy (%)	Kendall's τ	Spearman's ρ
Area-only	1.0	0.0	84.7 $\pm$ 2.9	0.69 $\pm$ 0.08	0.73 $\pm$ 0.07
Centroid-only	0.0	1.0	81.9 $\pm$ 3.2	0.64 $\pm$ 0.09	0.69 $\pm$ 0.08

Area + Centroid	0.8	0.2	89.6 ± 2.5	0.79 ± 0.07	0.83 ± 0.06
Area + Centroid	0.7	0.3	-92.1 ± 2.3	0.84 ± 0.06	0.87 ± 0.05
Proposed	0.6	0.4	93.8 ± 2.1	0.87 ± 0.05	0.90 ± 0.04
Area + Centroid	0.5	0.5	92.9 ± 2.2	0.86 ± 0.05	0.89 ± 0.04
Area + Centroid	0.4	0.6	90.8 ± 2.4	0.82 ± 0.06	0.86 ± 0.05
Area + Centroid	0.2	0.8	86.7 ± 2.8	0.74 ± 0.08	0.78 ± 0.07

When ranking based on the area of just one of the three objects, accuracy in pairwise ranking is about 84.7%. When ranking based on the centroid only, it is about 81.9%. Neither of these individual cues is enough for reliable ordinal estimation. The combination of the two cues improves ranking accuracy, with the highest accuracy (about 93.8% on the pairwise ranking) achieved at $\alpha = 0.6$ and $\beta = 0.4$. Kendall's τ for this combination is 0.87, and Spearman's ρ is 0.90. A reduction in performance with $\alpha < 0.6$ suggests that too much weight on the position of the centroid decreases the reliability of the ranking, with higher α values giving more weight to object size. The results support using the combined area–centroid formulation instead of either cue alone.

Table 12. Statistical Validation of the proposed relative depth estimation

Metric	Mean ± SD	95% CI
Pairwise Ranking Accuracy (%)	93.8 ± 2.1	93.33–94.27
Kendall's (τ)	0.87 ± 0.05	0.859–0.881
Spearman's (ρ)	0.90 ± 0.04	0.891–0.909

Mean ranking accuracy (μ) as shown in eqn 22 computes average performance over all images.

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i \quad (22)$$

Where,

- N = number of test images
- x_i = performance metric obtained for the image

Standard Deviation (SD) has been calculated based on eqn 23.

$$SD = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \mu)^2} \quad (23)$$

The 95% confidence interval (CI) depicts the range of the true average performance that is expected with 95% confidence, based on the test samples as shown in eqn 24.

$$CI_{95\%} \approx \mu \pm t_{0.975, N-1} \frac{SD}{\sqrt{N}} \quad (24)$$

The relative depth estimation accuracy was tested on 80 test images. Results shown in Table 12 are presented as mean ± SD and 95% CI was calculated using the student's t-distribution. For the proposed configuration ($\alpha=0.6$, $\beta=0.4$), the pairwise ranking accuracy was $93.8 \pm 2.1\%$,

with a 95% CI of 93.33–94.27%. Kendall's τ was 0.87 ± 0.05 (95% CI: 0.859–0.881), while Spearman's ρ was 0.90 ± 0.04 (95% CI: 0.891–0.909). It seems that the results show consistent Ordinal Depth-ranking performance on the test images measured.

5. Conclusion and Future Scope

In this paper, a comparative framework based on two different hardware approaches for relative and absolute depth estimation for resource-constrained embedded vision systems was proposed. The proposed relative depth estimation framework consists of image preprocessing, adaptive foreground segmentation, run-based Connected Component Labeling (CCL), and centroid-based object ranking to obtain ordinal depth information with low computational complexity. The absolute depth estimation framework, on the other hand, leverages a pre-trained Monodepth2 network to produce dense disparity maps and encodes the metric depth from the disparity maps using stereo camera geometry. This research compares both approaches within the same framework and shows the trade-offs between estimation accuracy, computational complexity, and hardware implementation cost. Experiments showed that the proposed run-based CCL has a computational complexity of $O(R)$, where R is the number of foreground runs, which results in fewer neighbor comparisons and fewer memory accesses than usual pixel-based labeling algorithms. The total FPGA run-time of the complete implementation for a 100×100 input image was about 30,000 clock cycles or 0.30 ms at 100 MHz. The usage of 3,460 LUTs, 4 BRAMs, and 0.78 W of dynamic power demonstrated that the relative depth estimation framework was suitable for real-time embedded vision applications. The CNN-based absolute depth estimation framework can achieve metric depth estimation accuracy on higher-resolution images in the KITTI (640 x 192) and NYU Depth V2 (640 x 480) datasets. This enhancement in depth accuracy comes at a high cost in terms of computational resources: around 18,000 LUTs, 28 BRAMs, 32 DSP blocks, and 1.80 W at runtime, which translates to around 22 FPS in practice. The results show that relative depth estimation is more appropriate for applications such as obstacle prioritization, collision avoidance, and mobile robotics, while absolute depth estimation is more suitable for applications requiring precise geometric scene reconstruction, autonomous navigation, and three-dimensional mapping. In summary, the comparative analysis is presented to offer practical guidelines for choosing the right technique to estimate the depth of a scene depending on application requirements, computational resources, and embedded hardware resources. The findings show that lightweight run-based image processing architectures are still very attractive for low-power FPGA-based embedded systems, whereas CNN-based architectures provide better metric depth estimation but at the cost of consuming more hardware resources.

Compliance with Ethical Standards

This study is not part of any funding. The authors declare no conflict of interest. This article does not contain any studies with human participants or animals performed by any of the authors.

References

- [1] Eigen, David, Christian Puhrsch, and Rob Fergus. "Depth Map Prediction from a Single Image Using a Multi-Scale Deep Network." *Advances in neural information processing systems* 27 (2014).

- [2] Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision Meets Robotics: The KITTI Dataset," *The International Journal of Robotics Research (IJRR)*, vol. 32, no. 11, 2013, 1231–1237.
- [3] Endsley, Mica R. "Design and Evaluation for Situation Awareness Enhancement." In *Proceedings of the Human Factors Society annual meeting*, vol. 32, no. 2, Sage CA: Los Angeles, CA: Sage Publications, 1988, 97-101.
- [4] Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised Monocular Depth Estimation with Left-Right Consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [5] Ranftl, René, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. "Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer." *IEEE transactions on pattern analysis and machine intelligence* 44, no. 3. 2020: 1623-1637.
- [6] Wu, Haifeng, Shuhang Gu, Lixin Duan, and Wen Li. "Geodepth: From Point-to-Depth to Plane-to-Depth Modeling for Self-Supervised Monocular Depth Estimation." In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2025, 11525-11535.
- [7] Liu, Fayao, Chunhua Shen, and Guosheng Lin. "Deep Convolutional Neural Fields for Depth Estimation from a Single Image." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, 5162-5170.
- [8] Godard, Clément, Oisín Mac Aodha, Michael Firman, and Gabriel J. Brostow. "Digging into self-supervised monocular depth estimation." In *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, 3828-3838.
- [9] Bhat, Shariq Farooq, Ibraheem Alhashim, and Peter Wonka. "Adabins: Depth estimation using adaptive bins." In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, 4009-4018.
- [10] Zoran, D., Isola, P., Krishnan, D. and Freeman, W.T., 2015. Learning Ordinal Relationships for Mid-Level Vision. In *Proceedings of the IEEE international conference on computer vision* (388-396).
- [11] Fu, Huan, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. "Deep Ordinal Regression Network for Monocular Depth Estimation." In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, 2018, 2002-2011.
- [12] Zhou, Hang, David Greenwood, and Sarah Taylor. "Self-Supervised Monocular Depth Estimation with Internal Feature Fusion." *arXiv preprint arXiv:2110.09482* (2021).
- [13] Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." *arXiv preprint arXiv:2010.11929* (2020).
- [14] Ranftl, René, Alexey Bochkovskiy, and Vladlen Koltun. "Vision Transformers for Dense Prediction." In *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, 12179-12188.
- [15] Yang, Lihe, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. "Depth Anything v2." *Advances in Neural Information Processing*

Systems 37 (2024): 21875-21911.

- [16] Wofk, Diana, Fangchang Ma, Tien-Ju Yang, Sertac Karaman, and Vivienne Sze. "Fastdepth: Fast Monocular Depth Estimation on Embedded Systems." In 2019 International Conference on Robotics and Automation (ICRA), IEEE, 2019, 6101-6108.
- [17] Zhang, Ning, Francesco Nex, George Vosselman, and Norman Kerle. "Lite-mono: A lightweight CNN and Transformer Architecture for Self-Supervised Monocular Depth Estimation." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2023, 18537-18546.
- [18] He, Lifeng, Yuyan Chao, and Kenji Suzuki. "A Run-Based Two-Scan Labeling Algorithm." IEEE transactions on image processing 17, no. 5 (2008): 749-756.
- [19] Di Stefano, Luigi, and Andrea Bulgarelli. "A Simple and Efficient Connected Components Labeling Algorithm." In Proceedings 10th international conference on image analysis and processing, IEEE, 1999, 322-327.
- [20] Grana, Costantino, Daniele Borghesani, and Rita Cucchiara. "Optimized Block-Based Connected Components Labeled with Decision Trees." IEEE Transactions on Image Processing 19, no. 6 (2010): 1596-1609.
- [21] Kiran, Divya, Abdul Imran Rasheed, and Hariharan Ramasangu. 2023. "Method, System and Apparatus for Object Detection." Indian Patent 466134 filed November 24, 2014 (Application No. 3573/CHE/2014) and issued November 6, 2023.
- [22] Kiran, Divya, Abdul Imran Rasheed, and Hariharan Ramasangu. "FPGA Implementation Of Blob Detection Algorithm for Object Detection in Visual Navigation." In 2013 International conference on Circuits, Controls and Communications (CCUBE), IEEE, 2013, 1-5.
- [23] He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask r-cnn." In Proceedings of the IEEE international conference on computer vision, 2017, 2961-2969.
- [24] Lavreniuk, Mykola, and Alla Lavreniuk. "Spidepth: Strengthened Pose Information for Self-Supervised Monocular Depth Estimation." In Proceedings of the Computer Vision and Pattern Recognition Conference, 2025, 874-884.
- [25] ul Islam, Nida, Divya Kiran, and Hariharan Ramasangu. "System-on-Chip Implementation of Situational Awareness System for Autonomous Cars." In 2018 15th IEEE India Council International Conference (INDICON), IEEE, 2018, 1-6.
- [26] Suda, Naveen, Vikas Chandra, Ganesh Dasika, Abinash Mohanty, Yufei Ma, Sarma Vrudhula, Jae-sun Seo, and Yu Cao. "Throughput-optimized OpenCL-based FPGA Accelerator for Large-Scale Convolutional Neural Networks." In Proceedings of the 2016 ACM/SIGDA international symposium on field-programmable gate arrays, 2016, 16-25.
- [27] Wang, Han, Xiaoyue Liu, Yi Yang, Qin Liao, Shiyu Ke, and Yong Dong. "FPGA-Based Acceleration of Deep Learning Networks: Techniques and Applications." In Proceedings of the 2025 6th International Conference on Computer Information and Big Data Applications, 2025, 774-778.
- [28] Obukhov, Anton, Matteo Poggi, Fabio Tosi, Ripudaman Singh Arora, Jaime Spencer, Chris Russel, Simon Hadfield et al. "The Fourth Monocular Depth Estimation Challenge."

- In Proceedings of the Computer Vision and Pattern Recognition Conference, 2025, 6182-6195.
- [29] Benkrid, Khaled, S. Sukhsawas, Danny Crookes, and Abdsamad Benkrid. "An FPGA-based Image Connected Component Labeller." In International Conference on Field Programmable Logic and Applications, Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, 1012-1015.
- [30] Spagnolo, Fanny, Fabio Frustaci, Stefania Perri, and Pasquale Corsonello. "An Efficient Connected Component Labeling Architecture for Embedded Systems." Journal of Low Power Electronics and Applications 8, no. 1 (2018): 7.
- [31] Perri, Stefania, Fanny Spagnolo, and Pasquale Corsonello. "A Parallel Connected Component Labeling Architecture for Heterogeneous Systems-on-Chip." Electronics 9, no. 2 (2020): 292.
- [32] Sada, Youki, Naoto Soga, Masayuki Shimoda, Akira Jinguji, Shimpei Sato, and Hiroki Nakahara. "Fast Monocular Depth Estimation on an FPGA." In 2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), IEEE, 2020, 143-146.
- [33] Hashimoto, Nobuho, and Shinya Takamaeda-Yamazaki. "FADEC: FPGA-based Acceleration of Video Depth Estimation by HW/SW Co-design." In 2022 International Conference on Field-Programmable Technology (ICFPT), IEEE, 2022, 1-9.
- [34] Li, Jie, Chuanlun Zhang, Heng Li, Shuangli Du, Wenxuan Yang, Xiaoyan Wang, and Yiguang Liu. "RE-LFDE: A Resource-Efficient Hardware Accelerator for Low-Bit Light Field Image Depth Estimation." ACM Transactions on Embedded Computing Systems 25, no. 3 (2026): 1-20.