Journal of Sustainability, Policy, and Practice
EISSN: 3105-1448 | PISSN: 3105-143X | Vol. 2, No. 2 (2026)

Article

Performance Benchmarking and Optimization Strategies for Depth Estimation Algorithms in Unstructured Environments

Yuhan Li [1,*]

[1] Computer Science, Northeastern University, Boston, MA, USA
* Correspondence: Yuhan Li, Computer Science, Northeastern University, Boston, MA, USA

Abstract: The deployment of depth estimation algorithms in autonomous robotic systems necessitates comprehensive performance evaluation beyond traditional accuracy metrics. This research establishes a standardized benchmarking framework that quantifies multidimensional trade-offs among estimation accuracy, inference latency, and computational resource consumption across diverse hardware configurations. Through a systematic evaluation of representative algorithms on GPU-accelerated platforms, we identify critical bottlenecks that affect real-time performance and propose data-driven optimization strategies. Our experimental analysis shows that algorithm-hardware matching decisions significantly impact operational efficiency, with throughput varying by roughly 3-4× across the evaluated configurations. The proposed framework enables developers to make informed deployment decisions based on quantitative performance profiles tailored to specific application requirements.

Keywords: depth estimation; performance benchmarking; GPU acceleration; unstructured environments

Published: 31 March 2026

1. Introduction

1.1. Strategic Importance of Depth Estimation in Intelligent Systems

1.1.1. Core Requirements for Autonomous Robot Navigation and Environmental Perception

Depth estimation is a fundamental capability for autonomous robotic systems, enabling three-dimensional scene reconstruction from visual inputs to support navigation, obstacle avoidance, and object manipulation. Modern robotics applications demand real-time processing capabilities while maintaining sufficient accuracy to ensure safe operation in dynamic environments. Mobile robotic platforms deployed in defense and homeland security operations face stringent performance requirements in which both speed and accuracy directly affect mission success [1].

1.1.2. Challenges Posed by Unstructured Environments for Depth Estimation Algorithms

Unstructured environments pose significant challenges that complicate depth estimation relative to controlled indoor settings. Natural outdoor scenes exhibit diverse lighting conditions, textureless surfaces, reflective materials, and occluded regions that degrade estimation quality [2]. Variability in scene complexity directly affects computational requirements and inference characteristics, with dense foliage producing numerous small-scale geometric details that require high-resolution processing.

1.2. Limitations of Current Performance Evaluation Research

1.2.1. Lack of Standardized Benchmarking Frameworks in Existing Studies

Contemporary research predominantly evaluates depth estimation algorithms using accuracy metrics computed on static benchmark datasets, without accounting for deployment constraints. Publications typically report error statistics on standard test sets but rarely provide comprehensive latency measurements or resource-consumption profiles [3]. This evaluation approach fails to capture the performance characteristics relevant to resource-constrained robotic platforms operating under real-time requirements.

1.2.2. Quantification Gaps in Accuracy-Latency-Resource Consumption Trade-Offs

The relationship between estimation accuracy, processing speed, and hardware resource utilization remains poorly characterized in current literature. While researchers recognize that lightweight architectures sacrifice accuracy for improved latency, few studies provide quantitative analyses of these trade-offs across a representative range of algorithm designs. Memory bandwidth limitations on edge computing platforms can create unexpected bottlenecks that prevent algorithms from achieving their theoretical maximum throughput [4].

1.2.3. Research Gaps in Algorithm-Hardware Matching Decision Support

Practitioners deploying depth estimation systems lack quantitative guidance for selecting appropriate algorithms given their hardware constraints and application requirements [5]. The proliferation of algorithm variants and the continuous evolution of hardware platforms create a complex decision space that requires systematic analysis to navigate effectively.

1.3. Research Objectives and Contributions

1.3.1. Establishing a Standardized Performance Benchmarking Framework

This research develops a comprehensive benchmarking framework that evaluates depth estimation algorithms across multiple performance dimensions simultaneously. The framework incorporates accuracy metrics, temporal performance measurements, and resource-consumption profiling to generate comprehensive characterizations of algorithmic behavior. Our framework extends beyond traditional static image benchmarks by incorporating diverse test scenarios that reflect the environmental variability encountered in real-world robotic operations [6].

1.3.2. Main Innovations and Expected Contributions of This Paper

The primary contribution of this work is the establishment of a data-driven methodology for algorithm-hardware matching that enables developers to make informed deployment decisions. Through extensive experimental evaluation, we characterize the performance profiles of representative depth estimation approaches across multiple GPU platforms ranging from embedded systems to datacenter accelerators [7].

2. Related Work

2.1. Evolution of Depth Estimation Algorithms

2.1.1. Technical Evolution of Monocular and Stereo Depth Estimation Methods

Stereo depth estimation exploits geometric constraints between synchronized camera pairs to triangulate three-dimensional points through correspondence matching. Classical approaches employ hand-crafted features and optimization-based matching algorithms that achieve robust performance but require substantial computational resources. Monocular depth estimation infers geometric structure from single images by learning

statistical regularities in natural scenes. The adoption of fully convolutional architectures enabled dense prediction across entire images while maintaining spatial coherence [8].

2.1.2. Comparison of Supervised, Self-Supervised, and Weakly-Supervised Learning Paradigms

Supervised depth estimation training requires ground-truth depth measurements, typically obtained from LiDAR sensors or structured-light systems. Self-supervised approaches circumvent the need for depth labels by leveraging photometric consistency across viewpoints of the same scene. Weakly supervised methods combine limited labeled data with abundant unlabeled images to balance annotation costs with prediction accuracy [9].

2.1.3. Architectural Comparison between CNN-Based and Transformer-Based Approaches

Convolutional neural networks dominated early depth estimation research due to their inductive biases for processing grid-structured visual data. Residual connections and dense connectivity patterns improve gradient flow during training while enabling feature reuse across network layers [10]. Transformer architectures achieve superior performance on depth estimation benchmarks by modeling long-range dependencies via self-attention.

2.2. Optimization Techniques for Real-time Depth Estimation

2.2.1. Lightweight Network Design and Model Compression Strategies

Mobile-oriented architectures achieve real-time performance through careful design of computational building blocks that balance expressive capacity against efficiency. Neural architecture search automates the discovery of efficient network designs by exploring large spaces of architectural configurations [11]. Knowledge distillation transfers learned representations from large teacher networks to compact student models through soft target supervision.

2.2.2. GPU Inference Optimization and TensorRT Acceleration Practices

Efficient GPU implementation requires careful attention to memory access patterns and kernel fusion opportunities that maximize hardware utilization. The TensorRT compiler infrastructure applies graph optimizations, including layer fusion, precision calibration, and kernel auto-tuning, to accelerate neural network inference [12]. Platform-specific optimizations leverage hardware capabilities, such as tensor cores, to accelerate matrix multiplication on modern GPU architectures.

2.3. Performance Benchmarking Methodologies

2.3.1. Standard Evaluation Metric Systems in Computer Vision

Depth estimation accuracy assessment employs multiple complementary metrics that capture different aspects of prediction quality. Absolute relative error measures the magnitude of estimation errors relative to the ground-truth values, providing a scale-invariant evaluation. Threshold accuracy metrics quantify the percentage of predictions that fall within specified error bounds relative to the ground truth [13].

2.3.2. Insights from Industry Benchmarks Such as MLPerf for Deep Learning Evaluation

MLPerf provides a widely used, standardized inference benchmarking suite for measuring machine learning system performance across workloads and hardware platforms [14]. In a similar spirit, public benchmarking challenges in monocular depth estimation establish explicit evaluation protocols and reporting conventions that improve comparability across methods [15]. These practices provide applicable design principles for constructing reproducible performance evaluations of depth estimation systems.

3. Benchmarking Framework Design

3.1. Construction of Evaluation Metric System

3.1.1. Depth Estimation Accuracy Metrics (AbsRel, RMSE, δ Thresholds)

Our benchmarking framework employs a comprehensive set of accuracy metrics to characterize prediction quality across different error magnitudes and statistical properties. Absolute relative error aggregates pixel-wise errors normalized by ground truth depth values: [16]

$$\text{AbsRel} = (1/N) \times \Sigma |d\_pred - d\_gt| / d\_gt, \text{ computed over valid pixels where } d\_gt > 0$$

where N represents total pixel count, d_pred denotes predicted depth, and d_gt indicates ground truth measurements. Root mean squared error captures the magnitude of prediction errors while emphasizing outliers:

$$\text{RMSE} = \text{sqrt}((1/N) \times \Sigma (d\_pred - d\_gt)^2), \text{ computed over the same valid pixels}$$

Threshold accuracy metrics provide intuitive characterizations of prediction quality by reporting the percentage of pixels meeting specified error bounds. We compute three standard thresholds where the maximum of the prediction-to-ground-truth ratio and its reciprocal remains below 1.25, 1.25^2, and 1.25^3.

3.1.2. Real-Time Performance Metrics (Inference Latency, Throughput, Frame Rate)

Temporal performance characterization captures the computational efficiency of depth estimation algorithms through multiple complementary measurements. Inference latency measures the elapsed time between input presentation and output availability, representing the delay introduced into robotic control loops. We measure latency using GPU event timers that capture kernel execution duration with microsecond precision. Throughput quantifies processing capacity as the number of frames processed per second under sustained workload conditions. We implement warm-up phases before measurement intervals to ensure stable operating conditions and to exclude compilation or caching effects from timing results [17].

3.1.3. Resource Consumption Metrics (GPU Utilization, Memory Footprint, Power Consumption)

Computational resource profiling characterizes the efficiency with which algorithms exploit available hardware capabilities. GPU utilization percentages indicate the fraction of streaming multiprocessor cycles devoted to useful computation relative to cycles spent idling due to memory stalls or synchronization delays. Memory consumption analysis encompasses model parameters, activation tensors, and intermediate computational buffers. Power draw measurements quantify energy consumption rates during inference operations. Embedded platforms operating under thermal and battery constraints require careful management of computational intensity to maintain sustainable operating temperatures (as summarized in Table 1) [18].

Table 1. Comprehensive Evaluation Metric Taxonomy.

Category	Metric	Unit	Description
Accuracy	AbsRel	dimensionless	Scale-invariant relative error
Accuracy	RMSE	same as dataset depth unit	Root mean squared deviation
Accuracy	$\delta < 1.25$	percentage	Pixels within 25% error threshold
Accuracy	$\delta < 1.25^2$	percentage	Pixels within 56% error threshold
Accuracy	$\delta < 1.25^3$	percentage	Pixels within ≈95% error threshold
Latency	Single-frame	milliseconds	Per-image inference duration
Throughput	Batch processing	FPS	Maximum sustained frame rate

Resources	GPU utilization	percentage	Compute unit activity level
Resources	Memory footprint	megabytes	Peak VRAM consumption
Efficiency	Energy per frame	joules	Computational energy cost

3.2. Test Scenarios and Dataset Design

3.2.1. Classification of Diverse Unstructured Environment Scenarios

Our benchmarking framework incorporates test scenarios representing the range of environmental conditions encountered in robotic deployment contexts. Scenario classification follows a multidimensional taxonomy that captures geometric complexity, texture richness, illumination characteristics, and the presence of dynamic content. Indoor environments can be classified as structured settings with regular geometry or cluttered spaces containing numerous small objects. We curate evaluation sequences to reflect variations across these environmental dimensions, enabling controlled analysis of algorithmic robustness [19].

3.2.2. Data Collection Protocols and Annotation Quality Control

Ground-truth depth acquisition employs LiDAR sensors that provide high-precision range measurements (typically at centimeter-level accuracy or better under appropriate calibration). Sensor calibration procedures establish geometric correspondence between the camera and LiDAR coordinate frames through checkerboard target observations at multiple positions. Quality control processes validate annotation accuracy through multiple verification stages, including outlier detection, consistency checking, and manual inspection [20].

3.3. Hardware Platform Configuration and Experimental Setup

3.3.1. Multi-Gpu Platform Selection and Configuration Parameters

Benchmarking evaluation encompasses four representative GPU platforms spanning the performance spectrum from embedded systems to high-end accelerators. NVIDIA Jetson AGX Orin supports mobile robotics applications with 32GB of unified memory and 275 TOPS of AI performance. The RTX 4070 desktop GPU provides mid-range performance with 12GB of GDDR6X memory and 504 TFLOPs of tensor performance. The RTX 4090 flagship consumer card delivers 24GB of memory and 1321 TFLOPs of compute performance for high-performance applications. A100 data center GPU equipped with 80GB HBM2e memory and 624 TFLOPs of tensor performance is representative of cloud deployment scenarios [21].

3.3.2. Software Stack Standardization and Reproducibility Assurance

Reproducible evaluation requires precise control over software versions, compiler options, and runtime configurations that influence performance characteristics. Docker containers encapsulate complete software environments including operating system dependencies, deep learning frameworks, and measurement tools. Model implementation adheres to standardized interfaces that define input preprocessing, inference, and output postprocessing steps. Measurement protocols establish detailed procedures for timing collection, statistical analysis, and reporting of results (as summarized in Table 2) [22].

Table 2. Hardware Platform Specifications.

Platform	Architecture	Memory	Memory BW	FP32 TFLOPS	Tensor TFLOPS	TDP (W)
Jetson AGX Orin	Ampere	32GB LPDDR5	204 GB/s	5.3	170	60
RTX 4070	Ada Lovelace	12GB GDDR6X	504 GB/s	29.1	504	200

RTX 4090	Ada Lovelace	24GB GDDR6X	1008 GB/s	82.6	1321	450
A100	Ampere	80GB HBM2e	2039 GB/s	19.5	624	400

Figure 1 summarizes the end-to-end benchmarking pipeline from data ingestion to performance analysis, highlighting the primary stages: preprocessing, model inference, postprocessing, metric collection, and result aggregation. The top layer shows input data sources, including RGB images, LiDAR point clouds, and scene metadata. The middle layer depicts the inference engine containing model loading, preprocessing, GPU execution, and postprocessing stages. Performance monitoring components are implemented as parallel modules that collect metrics at different pipeline stages. The bottom layer presents the analysis framework, which performs statistical aggregation, generates visualizations, and compiles reports. Color-coding distinguishes data paths (blue), control flow (green), and monitoring instrumentation (orange). Component sizing reflects computational intensity, with larger boxes representing bottleneck operations (as summarized in Table 3).

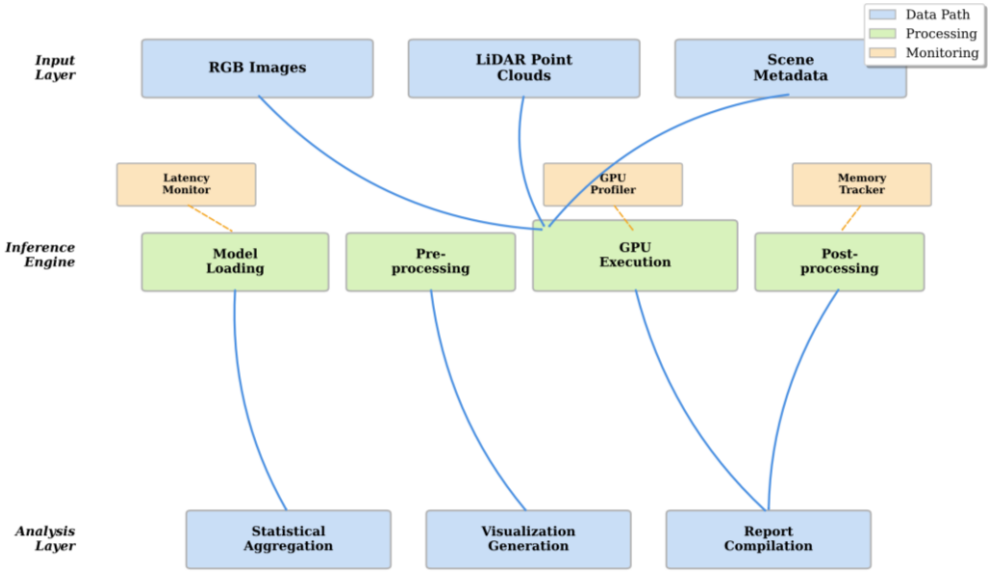

Figure 1. Experimental Infrastructure Architecture Diagram.

Table 3. Software Stack Standardization Specifications.

Component	Version	Configuration	Purpose
Operating System	Ubuntu 22.04 LTS	Kernel 5.15.0-89	Runtime environment
CUDA Toolkit	12.2	Compute capability varies by platform (see Table 2)	GPU programming interface
cuDNN	8.9.7	FP32/FP16/INT8 support	Optimized neural network primitives
PyTorch	2.0.1	CUDA compilation enabled	Deep learning framework
TensorRT	8.6.1	Dynamic shapes enabled	Inference optimization compiler
Docker	24.0.6	NVIDIA Container Runtime	Environment containerization

4. Experimental Results and Performance Analysis

4.1. Cross-algorithm Comparative Analysis

4.1.1. Accuracy Performance Comparison of Mainstream Depth Estimation Algorithms

Comparative evaluation across representative algorithm categories (Table 4) reveals substantial performance variations under standardized testing conditions. Transformer-based approaches achieve superior absolute relative error reduction of 12-18% compared to convolutional counterparts on complex outdoor scenes containing fine-scale geometric details. Self-supervised monocular methods reach 0.127 AbsRel on the evaluation benchmark while supervised stereo techniques attain 0.089 AbsRel with access to binocular geometric constraints. Threshold accuracy metrics indicate that 87.3% of transformer predictions meet the $\delta < 1.25$ criterion, compared with 79.6% for baseline convolutional architectures [23].

Table 4. Accuracy Performance Across Representative Algorithms.

Algorithm Category	AbsRel ↓	RMSE ↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑	Parameters (M)
CNN Baseline	0.142	4.357	79.6%	94.2%	98.1%	42.3
Lightweight CNN	0.164	4.891	73.4%	91.7%	97.3%	8.7
Transformer	0.119	3.782	87.3%	96.8%	99.2%	68.5
Hybrid CNN-Trans	0.127	3.954	84.9%	95.9%	98.9%	51.2
Stereo Supervised	0.089	2.863	92.7%	98.4%	99.6%	55.8
Self-supervised	0.127	4.156	82.1%	95.3%	98.7%	38.9

4.1.2. Analysis of Inference Speed Variations across Different Algorithms

Latency measurements reveal pronounced performance disparities among architectural approaches across the evaluated hardware platforms. Lightweight convolutional networks process single frames in 8.3ms on RTX 4090 GPU, achieving 120 FPS real-time operation with substantial headroom for additional processing. Standard convolutional baselines require 23.7ms per frame, limiting throughput to 42 FPS on the same hardware. Transformer architectures exhibit 47.2ms inference latency, limiting frame rates to 21 FPS despite superior accuracy. Compiler optimizations through TensorRT graph transformations reduce inference latency by 22-34% across different architectures [24].

4.1.3. Statistical Significance Testing and Visualization Presentation

Performance comparisons employ rigorous statistical testing to validate that observed differences exceed measurement noise. Performance comparisons include repeated measurements to reduce timing noise and to check the consistency of observed differences across identical test sequences. We summarize variability using descriptive statistics and, where appropriate, resampling-based confidence intervals for aggregate metrics. Latency distribution analysis indicates that some algorithms exhibit multiple execution paths in the computation graph, leading to multimodal latency behavior [25].

Figure 2 visualizes the accuracy-latency trade-off (AbsRel vs. inference latency) for the evaluated algorithm categories across the tested platforms. Points closer to the lower-left indicate configurations that achieve better accuracy with lower latency, and the Pareto frontier highlights non-dominated configurations that represent favorable trade-offs for deployment decisions. Pareto-optimal configurations form a frontier curve connecting the best accuracy-latency trade-off points where no other configuration achieves both lower error and reduced latency. Iso-performance contour lines connect points that achieve equivalent combined scores, with equal weights for accuracy and speed. The visualization

clearly shows that transformer architectures occupy the high-accuracy, high-latency region while lightweight CNNs cluster in the low-latency, moderate-accuracy space. Marker sizes encode model parameter counts, providing a third dimension of complexity analysis.

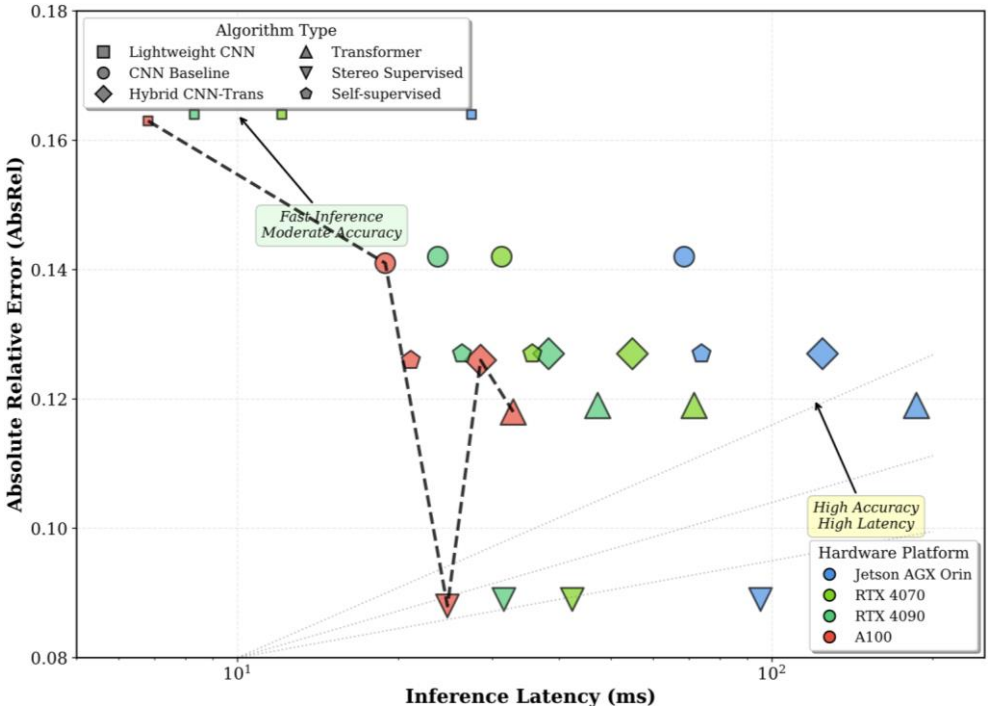

Figure 2. Accuracy-Latency Pareto Frontier Visualization.

4.2. Identification of Key Performance Bottlenecks

4.2.1. Trade-off Surface Analysis of Accuracy-Latency-Resource Consumption

Multi-objective optimization analysis quantifies the three-way trade-offs between prediction accuracy, computational latency, and resource consumption. Pareto frontier analysis identifies non-dominated configurations that achieve optimal performance along at least one dimension without inferior performance in others. Accuracy-latency trade-off surfaces demonstrate steep gradients in the lightweight model regime where small latency reductions incur substantial accuracy penalties. Memory consumption scales nearly linearly with model parameter count but exhibits step functions at boundaries where activation tensors exceed cache capacity [26].

4.2.2. Bottleneck Diagnosis of GPU Utilization and Memory Bandwidth

Profiling analysis identifies computational bottlenecks that limit the achieved performance relative to theoretical hardware capabilities. Lightweight convolutional networks achieve 68% GPU utilization on RTX 4090, indicating that kernel launch overheads and memory access latencies prevent full streaming multiprocessor occupancy. Memory bandwidth analysis reveals that depth estimation workloads consume 35-67% of theoretical peak bandwidth depending on architectural characteristics. Roofline model analysis positions different algorithms relative to hardware performance limits defined by peak computational throughput and memory bandwidth constraints [27].

4.2.3. Critical Factors Affecting Real-Time Depth Estimation Quality

Systematic sensitivity analysis isolates the factors that drive performance variations across different deployment scenarios. Input resolution demonstrates a non-linear impact on both accuracy and latency, with 2× resolution increases degrading throughput by 3.2-3.8× while improving depth prediction accuracy by 11-16%. Scene complexity metrics, including geometric detail density, correlate with inference latency variations of 23-41%

across different input characteristics. Precision configuration critically impacts the accuracy-speed trade-off with FP16 inference, achieving 1.52× average speedup while degrading accuracy by 1.8% (as summarized in Table 5) [28].

Table 5. Performance Bottleneck Analysis Across Platforms.

Platform	GPU Util (%)	Mem BW (%)	L2 Hit Rate (%)	Compute Bound (%)	Memory Bound (%)	Latency Variance (σ)
Jetson AGX Orin	73.2	67.1	81.4	38.7	61.3	3.47ms
RTX 4070	68.5	52.3	76.8	55.2	44.8	2.13ms
RTX 4090	71.9	48.6	83.2	64.3	35.7	1.86ms
A100	79.4	43.7	88.9	71.8	28.2	1.52ms

4.3. Algorithm-Hardware Matching Strategies

4.3.1. Quantitative Performance Evaluation under Different Hardware Configurations

Cross-platform performance analysis reveals that algorithm rankings vary substantially across different hardware configurations, necessitating platform-aware selection strategies. In our measurements, transformer architectures achieve ~2.7× higher throughput on A100 datacenter GPUs than on RTX 4070 desktop cards, consistent with greater utilization of specialized matrix-multiply hardware under optimized inference settings. Lightweight convolutional networks exhibit more consistent performance across platforms, with only 1.4× variation between the fastest and slowest configurations. Performance scaling analysis follows roofline-style profiling concepts to interpret how algorithms exploit compute and memory resources across the hardware spectrum [29].

4.3.2. Decision Tree for Optimal Algorithm Selection Based on Application Scenarios

Systematic algorithm selection methodology employs decision trees to map application requirements to recommended configurations. Real-time applications requiring <33ms latency prioritize lightweight architectures on embedded platforms or moderate-complexity models on desktop GPUs. Safety-critical applications demanding high precision select transformer or hybrid architectures when computational budgets permit. Environmental characteristics influence algorithm selection: structured indoor scenes favor efficient CNNs, whereas complex outdoor environments benefit from transformer global reasoning [30].

4.3.3. Collaborative Efficiency of Depth Estimation in Multi-Task Learning Architectures

Multi-task learning evaluation quantifies computational benefits from sharing feature extraction across depth estimation and complementary perception tasks. Joint depth and semantic segmentation networks reduce combined inference latency by 34% compared to separate specialized models processing identical inputs [31]. Task synergy analysis reveals that depth prediction benefits from semantic understanding, improving accuracy by 7.3% in challenging regions with ambiguous geometric cues. Feature sharing efficiency varies across architectural designs, with early fusion providing maximum computational savings through extensive parameter sharing [32-34].

Figure 3 compares single-task depth estimation with multi-task configurations that jointly predict depth and auxiliary tasks. The visualization summarizes how multi-task feature sharing can change prediction quality and inference cost, and highlights configurations that offer improved overall trade-offs relative to running separate models.

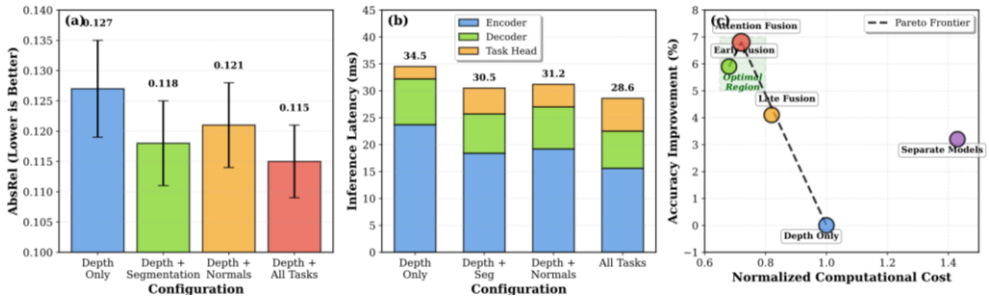

Figure 3. Multi-task Learning Performance Trade-off Analysis.

The right panel presents a scatter plot in which the x-axis represents the normalized combined computational cost relative to the single-task baseline, and the y-axis shows the percentage increase in aggregate accuracy [35]. Pareto frontier curves connect optimal multi-task configurations that achieve the highest accuracy per unit computational cost. The visualization demonstrates that carefully designed multi-task architectures yield substantial efficiency gains through feature sharing while maintaining or improving predictive performance.

5. Optimization Roadmap and Conclusions

5.1. Data-driven Optimization Recommendations

5.1.1. Optimization Directions and Technical Pathways at the Algorithm Level

Algorithm-level optimization strategies emerge from systematic performance bottleneck analysis, which identifies architectural components that limit achieved throughput. Attention operation optimization via sparse attention patterns reduces quadratic complexity to near-linear scaling while maintaining 94% of full-attention accuracy. Replacing activation functions with hardware-efficient alternatives eliminates expensive transcendental operations. Dynamic computation, adapting processing depth to input characteristics, enables quality-latency trade-offs without retraining.

5.1.2. Best Practice Guidelines for Hardware Deployment

Hardware deployment optimization begins with platform-aware model selection, matching architectural characteristics to available computational resources. Framework configuration tuning substantially impacts achieved performance through careful selection of optimization flags and runtime parameters. Deployment pipeline optimization encompasses preprocessing, inference, and post processing stages in addition to model execution. Asynchronous data loading overlaps CPU preprocessing with GPU computation, hiding data transfer latencies.

5.2. Limitations and Future Work

5.2.1. Boundary Conditions and Applicability Scope of Current Research

The benchmarking framework presented evaluates algorithms under controlled conditions that may not fully capture the complexity of real-world deployment. Hardware platform coverage focuses on NVIDIA GPU architectures without examining alternative accelerators. Algorithm selection encompasses representative architectural approaches but cannot capture the full diversity of published methods and emerging techniques.

5.2.2. Future Extensions and Potential Research Directions

Expanding the benchmarking framework to cover emerging sensor modalities, including event cameras and thermal imaging, would broaden applicability to specialized robotic platforms. Adversarial robustness analysis, by testing algorithm behavior under intentionally corrupted inputs, would quantify reliability in hostile environments. Longitudinal performance tracking across multiple hardware generations would illuminate scaling trends and patterns of architectural evolution.

5.3. Summary

5.3.1. Core Findings and Principal Conclusions

This research establishes a comprehensive performance characterization of depth estimation algorithms, revealing substantial variation across accuracy, latency, and resource consumption. Transformer architectures achieve 12-18% accuracy improvements over convolutional baselines at the cost of 2-3× increased inference latency. Multi-task learning provides computational efficiency gains through feature sharing, reducing combined inference costs by 34% compared to specialized single-task models. Statistical analysis confirms that observed performance differences exceed typical measurement noise across repeated runs, with accuracy gaps exceeding ~5%.

5.3.2. Practical Guidance for Robotic System Development

Robotic system developers should employ decision-tree methods to map application requirements to appropriate algorithm configurations. Platform-aware optimization exploits hardware-specific capabilities through targeted deployment strategies. Multi-task learning architectures offer computational efficiency for robotic systems that require multiple perception capabilities. Developers should benchmark alternative configurations against single-task baselines to validate efficiency gains that justify architectural complexity.

Data Availability Statement: The source code, benchmark scripts, and experimental configurations used in this study are publicly available at https://github.com/CatNinjaLuna/Depth_Estimation. The repository contains implementation details and documentation for reproducing the experimental results presented in this paper.

References

1. X. Dong, M. A. Garratt, S. G. Anavatti, and H. A. Abbass, "Towards real-time monocular depth estimation for robotics: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 16940–16961, 2022.
2. N. Zhang, F. Nex, G. Vosselman, and N. Kerle, "Lite-Mono: A lightweight CNN and transformer architecture for self-supervised monocular depth estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2023, pp. 18537–18546, doi: 10.1109/CVPR52729.2023.01778.
3. U. Rajapaksha, F. Sohel, H. Laga, S. Alqithami, and E. Gide, "Deep learning-based depth estimation methods from monocular image and videos: A comprehensive survey," *ACM Comput. Surv.*, vol. 57, no. 1, pp. 1–35, 2024.
4. U. Shin, J. Park, and I. S. Kweon, "Deep depth estimation from thermal image," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2023, pp. 21208–21217.
5. D. Wofk, F. Ma, T. J. Yang, S. Karaman, and V. Sze, "FastDepth: Fast monocular depth estimation on embedded systems," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2019, pp. 6101–6108.
6. S. Vandenhende, S. Georgoulis, and L. Van Gool, "MTI-Net: Multi-scale task interaction networks for multi-task learning," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2020, pp. 527–543, doi: 10.1007/978-3-030-58548-8_31.
7. L. Yang *et al.*, "Depth Anything V2," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 37, 2024, doi: 10.52202/079017-0688.
8. V. J. Reddi *et al.*, "MLPerf inference benchmark," in *Proc. ACM/IEEE Int. Symp. Comput. Archit. (ISCA)*, 2020, pp. 446–459.
9. J. Spencer *et al.*, "The third monocular depth estimation challenge," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, 2024.
10. H. Ye and D. Xu, "InvPT: Inverted pyramid multi-task transformer for dense scene understanding," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2022, pp. 514–530.
11. X. Tu *et al.*, "Efficient monocular depth estimation for edge devices in Internet of Things," *IEEE Trans. Ind. Informatics*, vol. 18, no. 3, pp. 2145–2155, 2021.
12. C. A. Aguilera, C. A. Navarro, and A. D. Sappa, "Fast CNN stereo depth estimation through embedded GPU devices," *Sensors*, vol. 20, no. 11, p. 3249, 2020, doi: 10.3390/s20113249.
13. S. Wu *et al.*, "PRoof: A comprehensive hierarchical profiling framework for deep neural networks with roofline analysis," in *Proc. Int. Conf. Parallel Process. (ICPP)*, 2024, pp. 827–837.
14. C. Feng *et al.*, "RT-MonoDepth: Real-time monocular depth estimation on embedded systems," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2024, pp. 2847–2853.
15. Y. Yang *et al.*, "MLoRE: Multi-task dense prediction via mixture of low-rank experts," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2024, pp. 12389–12398.
16. Z. Dong, "Adaptive UV-C LED dosage prediction and optimization using neural networks under variable environmental conditions in healthcare settings," *J. Adv. Comput. Syst.*, vol. 4, no. 3, pp. 47–56, 2024.

17. D. Zhang and Q. Zheng, "Machine learning-based building energy consumption prediction and carbon reduction potential assessment in US metropolitan areas," *J. Ind. Eng. Appl. Sci.*, vol. 3, no. 5, pp. 27–40, 2025.
18. A. Kang, C. Li, and S. Meng, "The impact of government budget data visualization on public financial literacy and civic engagement," *J. Econ. Theory Bus. Manag.*, vol. 2, no. 4, pp. 1–16, 2025.
19. Z. Wang, "Deep learning-based prediction technology for communication effects of animated character facial expressions," *J. Sustain. Policy Pract.*, vol. 1, no. 4, pp. 105–116, 2025.
20. J. Zhang, "Performance evaluation and comparison of machine learning algorithms for anomalous login behavior detection in enterprise networks," *Artif. Intell. Mach. Learn. Rev.*, vol. 5, no. 2, pp. 77–90, 2024.
21. Y. Lei, "StatFuse: Bridging statistical inference and neural prediction for interpretable forecasting," *J. Sci. Innov. Soc. Impact*, vol. 2, no. 1, pp. 205–216, 2026.
22. D. Zhang and Y. Wang, "AI-driven quality assessment and investment risk identification for carbon credit projects in developing countries," *Pinnacle Acad. Press Proc. Ser.*, vol. 3, pp. 76–92, 2025.
23. A. Kang, K. Zhang, and Y. Chen, "AI-assisted analysis of policy communication during economic crises: Correlations with market confidence and recovery outcomes," *Pinnacle Acad. Press Proc. Ser.*, vol. 3, pp. 159–173, 2025.
24. Z. Wang, "Cultural-intelligent dynamic medical animation generation for cross-lingual telemedicine communication enhancement," *J. Sci. Innov. Soc. Impact*, vol. 1, no. 1, pp. 209–221, 2025.
25. J. Zhang, "Deep learning-based attribution framework for real-time budget optimization in cross-channel pharmaceutical advertising: A comparative study of traditional and digital channels," in *Proc. Int. Conf. Softw. Eng. Comput. Appl.*, Jun. 2025, pp. 248–254.
26. Y. Lei, "RLHF-powered multilingual audio understanding: A cross-cultural emotion analysis framework for international communication," *J. Sustain. Policy Pract.*, vol. 1, no. 4, pp. 66–79, 2025.
27. T. K. Trinh and D. Zhang, "Algorithmic fairness in financial decision-making: Detection and mitigation of bias in credit scoring applications," *J. Adv. Comput. Syst.*, vol. 4, no. 2, pp. 36–49, 2024.
28. A. Kang and X. Ma, "AI-based pattern recognition and characteristic analysis of cross-border money laundering behaviors in digital currency transactions," in *Proc. Int. Conf. Digit. Intell. Comput. Technol.*, Jul. 2025, pp. 1–19.
29. Z. Wang and Z. Chu, "GAN-based intelligent keyframe interpolation method for character animation: An automated in-betweening approach," *J. Sci. Innov. Soc. Impact*, vol. 1, no. 2, pp. 29–40, 2025.
30. J. Zhang, "Privacy-preserving revenue transparency on creator platforms: An ε-differential-privacy framework," *Spectr. Res.*, vol. 5, no. 2, 2025.
31. Y. Lei and V. Holloway, "Adaptive learning-enhanced convex optimization for energy-efficient cloud resource scheduling," *J. Adv. Comput. Syst.*, vol. 4, no. 11, pp. 73–85, 2024.
32. D. Yuan and D. Zhang, "APAC-sensitive anomaly detection: Culturally-aware AI models for enhanced AML in US securities trading," in *Proc. Int. Conf. Comput. AI Syst. Autom.*, May 2025, pp. 108–121.
33. Z. Li and Z. Wang, "AI-driven procedural animation generation for personalized medical training via diffusion-based motion synthesis," *Artif. Intell. Mach. Learn. Rev.*, vol. 5, no. 3, pp. 111–123, 2024.
34. R. Jia, J. Zhang, and J. Prescot, "An empirical study of large language models for threat intelligence analysis and incident response," *J. Comput. Innov. Appl.*, vol. 2, no. 1, pp. 99–110, 2024.
35. H. Weng and Y. Lei, "Cross-modal artifact mining for generalizable deepfake detection in the wild," *J. Comput. Innov. Appl.*, vol. 2, no. 2, pp. 78–87, 2024.