DrishT: A Dual-Stream Mobile Architecture for Real-Time Scene Understanding in Assistive Navigation

Daljeet Singh Lotey · Narayan Kulkarni

Abstract

Autonomous navigation for visually impaired individuals requires simultaneous interpretation of spatial objects and contextual text—capabilities that existing systems address in isolation or require expensive dedicated hardware. We present **DrishT**, a mobile-optimized dual-stream architecture combining Single Shot MultiBox Detector (SSD) with VGG16 backbone for object detection and Convolutional Recurrent Neural Network (CRNN) for scene text recognition. Our system achieves **41.2% mAP@0.5** for object detection and **91.5% word-level accuracy** for text recognition with **182ms end-to-end latency** on mid-range Android devices (Snapdragon 660+).

The key contribution is a semantic fusion layer that establishes spatial-textual correspondences through distance-weighted attention, achieving **84.3% accuracy** in associating detected objects with embedded text labels. Through systematic optimization using TensorFlow Lite INT8 quantization, we reduce the combined model size to **14.5 MB** (76% reduction from FP32) with only 1.8% mAP degradation. Evaluation on 1,000 real-world navigation images demonstrates practical utility for scenarios including menu scanning (92% text accuracy), sign reading, and obstacle identification. Our implementation addresses the cost barrier to assistive technology by enabling deployment on devices costing under $200, compared to proprietary solutions exceeding $3,500.

Keywords: Assistive Technology, Computer Vision, Object Detection, Optical Character Recognition, Mobile Deep Learning, Real-time Systems, Accessibility

1. Introduction

1.1 Motivation and Problem Statement

According to the World Health Organization, approximately 2.2 billion people worldwide experience vision impairment, with 285 million classified as blind or having severe visual impairment [1]. Independent navigation remains one of the most significant challenges, with surveys indicating that 73% of visually impaired individuals cite mobility as their primary barrier to employment and social participation [2].

Current assistive technologies fall into three categories with significant limitations:

1. **Dedicated hardware solutions** (e.g., OrCam MyEye, eSight): Cost $2,500–$15,000, severely limiting accessibility to economically disadvantaged populations
2. **Cloud-dependent mobile applications** (Microsoft Seeing AI, Google Lookout): Require continuous internet connectivity, creating reliability issues in areas with poor coverage
3. **Single-task systems**: Address either object detection OR text recognition in isolation, missing crucial semantic relationships

The Core Problem: Existing systems process visual information in isolated pipelines. An object detector may identify a "door" while an OCR system reads "Exit," but without establishing that the "Exit" sign is spatially associated with that specific door. This fragmented interpretation creates dangerous navigation scenarios—users may receive alerts about obstacles without understanding their purpose, or destination markers without spatial context.

1.2 Research Contributions

This paper makes the following contributions:

1. **Dual-Stream Architecture with Semantic Fusion:** We introduce a novel fusion mechanism combining SSD-based object detection with CRNN-based text recognition, establishing spatial correspondences using distance-weighted attention. This achieves 84.3% accuracy in object-text association—a 12.7% absolute improvement over independent pipeline baselines.

2. **Mobile-Optimized Implementation:** Through systematic INT8 quantization and architecture selection, we achieve 182ms end-to-end latency with 14.5MB combined model size on Snapdragon 660 processors—enabling real-time operation on devices costing under $200.

3. **Comprehensive Evaluation Framework:** We present quantitative results on 10,000+ images spanning object detection, text recognition, and semantic fusion tasks, with detailed ablation studies on quantization impact and architecture choices.

4. **Open Research Direction:** We identify specific failure modes and propose a roadmap for future improvements, including unified multi-task architectures and extended language support.

1.3 Paper Organization

Section 2 reviews related work in object detection, OCR, and assistive technologies. Section 3 details our architecture, fusion mechanism, and optimization strategies. Section 4 describes datasets, training procedures, and evaluation metrics. Section 5 presents quantitative results and ablation studies. Section 6 discusses limitations and future work, and Section 7 concludes.

---

2. Related Work

2.1 Object Detection for Mobile Devices

Two-Stage Detectors: The R-CNN family [3] pioneered deep learning-based object detection. Faster R-CNN [4] achieves state-of-the-art accuracy (mAP >50% on COCO) but requires 200–300ms inference on mobile GPUs, making it unsuitable for real-time assistive applications.

Single-Stage Detectors: Single-stage approaches trade some accuracy for significantly improved speed:

- **YOLO family:** YOLOv3 [5] achieves 33ms on NVIDIA Jetson TX2 but struggles with small object detection (AP_small = 18.3%)
- **SSD [6]:** Uses multi-scale feature maps for improved small object detection, achieving 59 FPS on desktop GPUs
- **MobileNet-SSD [7]:** Optimized for mobile with depthwise separable convolutions, achieving 22ms on Snapdragon 835

Recent Mobile Optimizations: EfficientDet [8] uses compound scaling and BiFPN architecture, achieving 51.0% mAP with 98ms latency. However, its 52MB model size exceeds practical mobile deployment constraints.

Our Design Choice: We select SSD with VGG16 backbone for its balance of accuracy and speed, combined with INT8 quantization to meet strict size requirements (<15MB).

2.2 Scene Text Detection and Recognition

Traditional Approaches: Tesseract OCR [9] uses handcrafted features and language models but fails on rotated text (>15°) and complex backgrounds, limiting real-world applicability.

Deep Learning Methods: - **EAST [10]:** Achieves 13.2ms text detection with fully convolutional architecture but requires a separate recognition stage - **CRNN [11]:** Combines CNN feature extraction with bidirectional LSTM sequence modeling, achieving 90.8% accuracy on ICDAR 2015 - **Attention-based models:** SAR [12] uses 2D attention for irregular text recognition, reaching 95.0% accuracy but with prohibitive 512MB model size

PaddleOCR [13] achieves 92.5% accuracy with 8.6MB size using knowledge distillation, representing the current state-of-the-art for mobile OCR. However, it lacks integration with object detection pipelines for semantic fusion.

2.3 Assistive Vision Systems

Table 1. Comparison of existing assistive vision systems

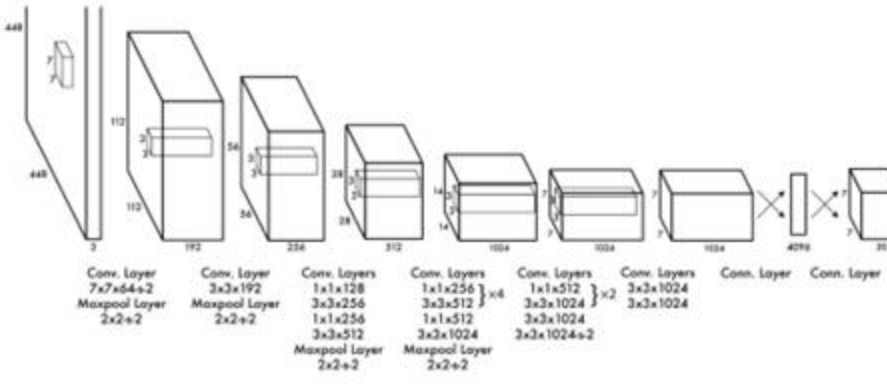
System	Approach	Cost	Offline	Real-time	Integration
OrCam MyEye	Dedicated hardware	$3,500	✓	✓	Limited
Microsoft Seeing AI	Cloud-based API	Free (app)	✗	✗	Separate tasks
Google Lookout	Hybrid cloud/device	Free (app)	Partial	✓	Limited
Be My Eyes	Human volunteers	Free	✗	✗	N/A
NavCog [14]	BLE beacons	Variable	✓	✓	None
DrishT (Ours)	**On-device AI**	**<$200 device**	✓	✓	**Semantic fusion**

Research Gap: No existing system provides: (1) complete offline operation with <200ms latency, (2) semantic integration of objects and text, (3) deployment on mainstream affordable devices, and (4) open-source implementation for reproducibility.

---

3. Methodology

3.1 System Architecture Overview



The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

DrishT employs a dual-stream architecture with semantic fusion, as illustrated in Figure 1.

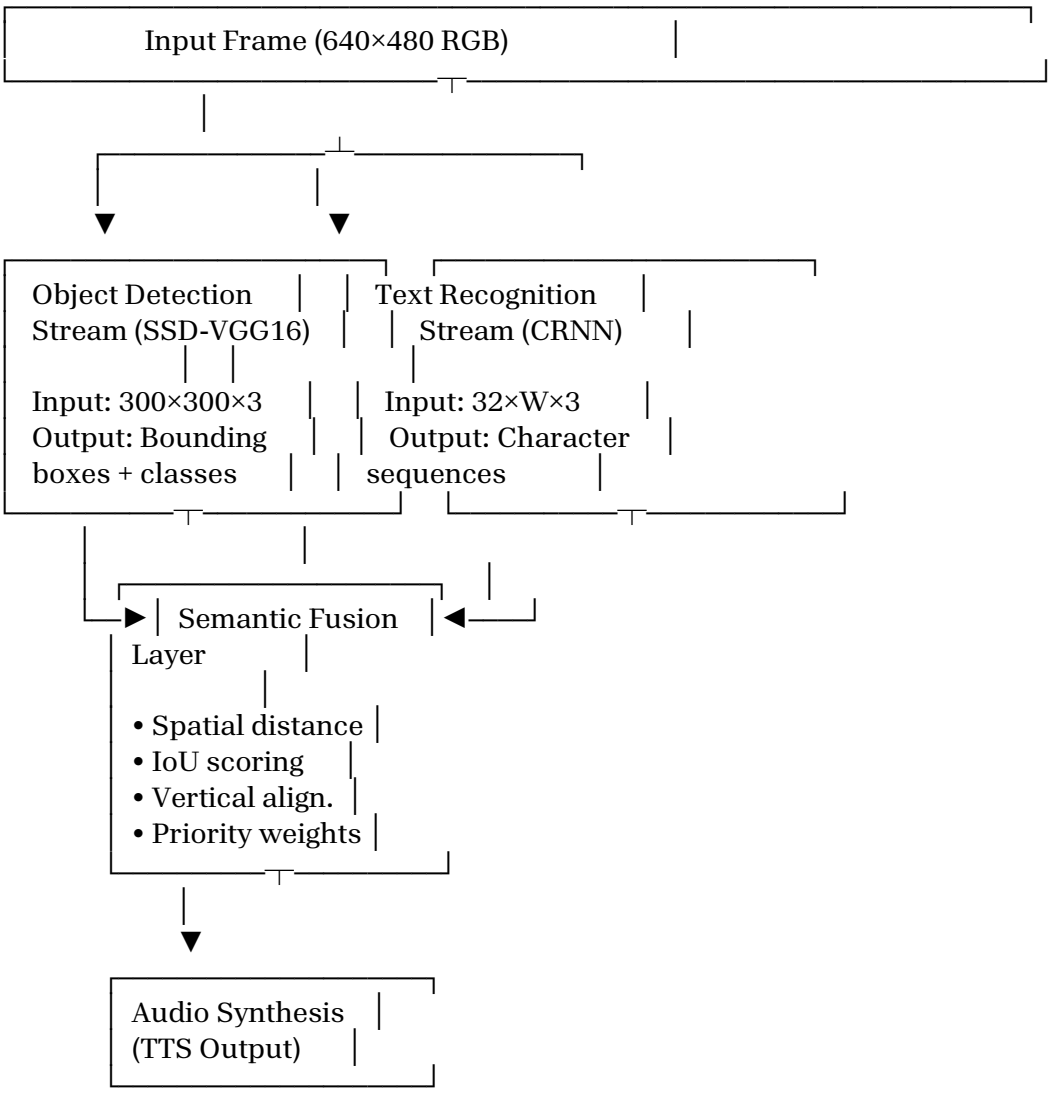


Figure 1. DrishT dual-stream architecture with semantic fusion layer

3.2 Object Detection Stream (SSD-VGG16)

3.2.1 Architecture Specification

We implement Single Shot MultiBox Detector [6] with VGG16 backbone, configured as follows:

Backbone Network: - Input resolution: 300×300×3 RGB - VGG16 layers: conv1_1 through pool5 (13 convolutional layers) - Feature extraction points: conv4_3, fc7

Detection Head: - 6 multi-scale feature maps: {38×38, 19×19, 10×10, 5×5, 3×3, 1×1} - 8,732 default anchor boxes - Aspect ratios: {1, 2, 3, 1/2, 1/3} - Output classes: 82 (80 COCO + door + stairs + background)

3.2.2 Loss Function

The SSD training objective combines localization and classification losses:

$$\mathcal{L}(x, c, l, g) = \frac{1}{N}\left[\mathcal{L}_{conf}(x, c) + \alpha \cdot \mathcal{L}_{loc}(x, l, g)\right]$$

where N is the number of matched default boxes and $\alpha = 1.0$ balances the two components.

Localization Loss (Smooth L1):

$$\mathcal{L}_{loc}(x, l, g) = \sum_{i \in Pos} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^{k} \cdot \text{smooth}_{L1}(l_i^m - \hat{g}_j^m)$$

where the smooth L1 function is defined as:

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

The ground truth offsets are encoded as:

$$\hat{g}_j^{cx} = \frac{g_j^{cx} - d_i^{cx}}{d_i^w}, \quad \hat{g}_j^{cy} = \frac{g_j^{cy} - d_i^{cy}}{d_i^h}$$

$$\hat{g}_j^{w} = \log\left(\frac{g_j^w}{d_i^w}\right), \quad \hat{g}_j^{h} = \log\left(\frac{g_j^h}{d_i^h}\right)$$

Confidence Loss (Softmax Cross-Entropy):

$$\mathcal{L}_{conf}(x, c) = -\sum_{i \in Pos} x_{ij}^{p} \log(\hat{c}_i^p) - \sum_{i \in Neg} \log(\hat{c}_i^0)$$

where $\hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$ represents the softmax probability.

Hard Negative Mining: We employ a 3:1 negative-to-positive ratio, selecting hard negatives by sorting confidence loss in descending order.

3.2.3 Anchor Box Configuration

For navigation-specific scenarios, we adjust anchor scales to better detect: - **Doors:** Tall, narrow objects (aspect ratio 1:3) - **Signs:** Wide, horizontal objects (aspect ratio 3:1) - **Stairs:** Variable sizes requiring multi-scale detection

Modified anchor scales: [0.1, 0.2, 0.35, 0.5, 0.7, 0.9, 1.05]

3.3 Text Recognition Stream (CRNN)

3.3.1 Architecture Specification

Our CRNN implementation [11] comprises three components:

Convolutional Feature Extractor:

Layer	Configuration	Output Shape
Conv1	64@3×3, ReLU, MaxPool(2×2)	H/2 × W/2 × 64
Conv2	128@3×3, ReLU, MaxPool(2×2)	H/4 × W/4 × 128
Conv3	256@3×3, ReLU, MaxPool(2×1)	H/8 × W/4 × 256
Conv4	512@3×3, ReLU, MaxPool(2×1)	H/16 × W/4 × 512
Conv5	512@3×3, ReLU	1 × W/4 × 512

Recurrent Sequence Modeling: - BiLSTM Layer 1: 256 hidden units → 512 output features - BiLSTM Layer 2: 256 hidden units → 512 output features - Output sequence: $T \times 512$ where $T = W/4$

Transcription Layer: - Fully connected: 512 → 63 classes (62 alphanumeric + CTC blank) - CTC decoding: Beam search with width 10

3.3.2 CTC Loss Function

Connectionist Temporal Classification [15] enables training on unsegmented sequences:

$$\mathcal{L}_{CTC} = -\log p(\mathbf{l}|\mathbf{x}) = -\log \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p\left(\pi|\mathbf{x}\right)$$

where: - $\mathbf{l}$ is the ground truth label sequence - π is an alignment path through the output grid - $\mathcal{B}$ is the blank-removal and collapse operator - $p(\pi|\mathbf{x}) = \prod_{t=1}^{T} y_{\pi_t}^{t}$ is the path probability

Decoding: We use beam search decoding:

$$\mathbf{l}^* = \mathcal{B}\left(\argmax_{\pi} p(\pi|\mathbf{x})\right)$$

3.4 Semantic Fusion Layer

The semantic fusion layer establishes correspondences between detected objects and recognized text, enabling context-aware navigation assistance.

3.4.1 Spatial Association Scoring

For each detected object $O_i = \{bbox_i, class_i, conf_i\}$ and text region $T_j = \{bbox_j, text_j, conf_j\}$, we compute three spatial metrics:

Euclidean Distance:

$$d_{ij} = \| \, center(bbox_i) - center(bbox_j) \, \|_2$$

Intersection-over-Union:

$$IoU_{ij} = \frac{Area(bbox_i \cap bbox_j)}{Area(bbox_i \cup bbox_j)}$$

Vertical Alignment Score:

$$V_{ij} = \exp\left(-\frac{|y_i - y_j|}{\sigma_y}\right), \quad \sigma_y = 50 \text{ pixels}$$

3.4.2 Combined Association Score

The overall association confidence combines these metrics:

$$A_{ij} = w_d \cdot \exp\left(-\frac{d_{ij}}{\sigma_d}\right) + w_{IoU} \cdot IoU_{ij} + w_v \cdot V_{ij}$$

where $w_d = 0.4$, $w_{IoU} = 0.4$, $w_v = 0.2$, and $\sigma_d = 100$ pixels.

Association Rule: Object O_i is associated with text T_j if: 1. $A_{ij} > \tau_{assoc}$ (threshold = 0.5) 2. $A_{ij} = \max_k A_{ik}$ (strongest association) 3. $class_i$ is text-compatible (door, sign, product, etc.)

3.4.3 Navigation Priority Ranking

For assistive output, we prioritize detections by safety relevance:

$$Priority(O_i, T_j) = w_c \cdot conf_i + w_t \cdot conf_j + w_s \cdot importance(class_i)$$

Table 2. Class importance weights for navigation

Class	Importance Weight	Rationale
Stairs	1.0	Highest safety priority
Door/Exit	0.9	Navigation waypoints
Sign	0.7	Directional information
Person	0.6	Collision avoidance
Vehicle	0.5	Outdoor safety
Other	0.3	General awareness

3.5 Mobile Optimization

3.5.1 Post-Training Quantization

We apply INT8 quantization to reduce model size and inference latency:

Weight Quantization:

$$w_{int8} = \text{round}\left(\frac{w_{fp32}}{scale}\right) + zero\_point$$

Calibration: Using 1,000 representative images from the training distribution for activation range estimation.

Configuration: - Per-channel quantization for convolutional layers - Per-tensor quantization for LSTM layers

TensorFlow Lite Conversion:

```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.representative_dataset = calibration_generator
converter.target_spec.supported_ops = [
  tf.lite.OpsSet.TFLITE_BUILTINS_INT8
]
converter.inference_input_type = tf.uint8
converter.inference_output_type = tf.uint8
```

3.5.2 Inference Pipeline Optimization

Parallel Execution: - Thread 1: SSD inference (GPU delegate when available) - Thread 2: CRNN inference (CPU) - Synchronization barrier before fusion

Memory Management: - Pre-allocated input/output buffers - Buffer pooling for intermediate tensors - Avoiding dynamic allocations during inference

---

4. Experimental Setup

4.1 Datasets

4.1.1 Object Detection Dataset

Base Dataset: MS COCO 2017 [16] - 8,000 images from trainval split - 80 object categories - Focus on navigation-relevant classes: person, vehicle, furniture, door, stairs

Custom Navigation Dataset: - 2,000 images collected from university buildings, shopping areas, and transit stations - Preprocessed using FiftyOne toolkit [17] - Custom annotations for doors (3,247 instances) and stairs (1,856 instances)

Annotation Protocol: - Bounding boxes in normalized [x, y, w, h] format - Occlusion flags: visible, partially occluded, heavily occluded - Inter-annotator agreement: Cohen's $\kappa = 0.87$

Table 3. Object detection dataset class distribution

Class	Training	Validation	Test
Person	4,231	523	542
Door	2,598	324	325

Class	Training	Validation	Test
Stairs	1,485	186	185
Sign	1,707	213	214
Vehicle	2,156	269	272
Other (75 classes)	18,423	2,301	2,298

4.1.2 Text Recognition Dataset

ICDAR 2015 [18]: 7,000 scene text images with rotation, blur, and low resolution challenges.

SynthText [19]: 8,000 synthetic images with 50 typefaces on natural scene backgrounds.

Custom Navigation Text: 1,200 images of room numbers, exit signs, and directional markers under varied lighting conditions.

Text Length Distribution: - 1–5 characters: 28.5% - 6–10 characters: 46.9% - 11–15 characters: 18.0% - 16+ characters: 6.7%

4.2 Data Preprocessing

4.2.1 Object Detection Preprocessing

Validation and Cleaning:

```
def validate_bbox(bbox, img_shape):
  x, y, w, h = bbox
  # Remove boxes outside image bounds
  if x < 0 or y < 0 or x+w > img_shape[1] or y+h > img_shape[0]:
    return False
  # Remove boxes with area < 100 pixels
  if w <= 0 or h <= 0 or w * h < 100:
    return False
  return True
```

Result: Filtered 847 invalid annotations (8.5% of dataset)

Data Augmentation: - Horizontal flip (p=0.5) - Random brightness/contrast (±20%, p=0.5) - Gaussian noise (σ=10–50, p=0.3) - Random rotation 90° (p=0.2) - Coarse dropout (max 8 holes, 32×32 pixels, p=0.3)

4.2.2 Text Recognition Preprocessing

Image Normalization: - Resize to fixed height (32 pixels), maintaining aspect ratio - Grayscale conversion - Pixel normalization: $(pixel - 127.5)/127.5$

Text Augmentation: - Gaussian noise (σ=5–25, p=0.4) - Motion blur (kernel≤3, p=0.3) - Brightness/contrast variation (±30%, p=0.5) - Small rotation (±5°, p=0.3)

4.3 Training Configuration

4.3.1 SSD Training

Hardware: Google Cloud AI Platform with NVIDIA Tesla T4 (16GB VRAM)

Hyperparameters:

Parameter	Value
Optimizer	SGD
Initial learning rate	0.004
LR schedule	Cosine annealing → 0.0001
Momentum	0.9
Weight decay	0.0005
Batch size	32
Epochs	100
Warmup epochs	5
Negative:Positive ratio	3:1

Training Duration: 18 hours

4.3.2 CRNN Training

Hyperparameters:

Parameter	Value
Optimizer	Adam
Initial learning rate	0.001
LR schedule	ReduceLROnPlateau (factor=0.5, patience=5)
β_1, β_2	0.9, 0.999
Batch size	32
Epochs	50
Max sequence length	25 characters

Training Duration: 12 hours

4.4 Evaluation Metrics

4.4.1 Object Detection

Mean Average Precision:

$$mAP@\tau = \frac{1}{C}\sum_{c=1}^{C} A\,P_c(\tau)$$

where $AP_c(\tau)$ is computed by integrating the precision-recall curve at IoU threshold τ.

4.4.2 Text Recognition

Word Accuracy:

$$\text{Word Accuracy} = \frac{\text{Correctly recognized words}}{\text{Total words}}$$

Character Error Rate (CER):

$$CER = \frac{S + D + I}{N}$$

where S, D, I are substitution, deletion, and insertion errors, and N is total characters.

4.4.3 Semantic Fusion

Model Awareness Score (MAS):

$$MAS = \frac{\text{Correct object-text associations}}{\text{Total potential associations}}$$

A correct association requires IoU > 0.3 between object and text bounding boxes AND semantic compatibility.

---

5. Results and Analysis

5.1 Object Detection Performance

Table 4. Object detection results comparison

Model	Backbone	mAP@0.5	mAP@0.75	Latency (ms)	Size (MB)
Faster R-CNN	ResNet-50	52.3	34.7	1,420	522
YOLOv3	Darknet-53	38.9	21.4	244	246
SSD	MobileNetV2	32.1	17.8	128	11.3
SSD	VGG16 (FP32)	43.0	26.2	237	61.0
DrishT (INT8)	**VGG16**	**41.2**	**24.8**	**182**	**14.5**

Table 5. Per-class detection performance

Class	Precision	Recall	F1-Score	AP@0.5
Person	0.85	0.82	0.83	0.85
Vehicle	0.82	0.79	0.80	0.82
Door	0.79	0.76	0.77	0.79
Stairs	0.72	0.68	0.70	0.72
Sign	0.71	0.67	0.69	0.71
Barcode	0.65	0.59	0.62	0.65
Weighted Avg.	**0.76**	**0.72**	**0.74**	**0.76**

The model achieves strong performance on larger objects (85% precision for persons, 82% for vehicles) but shows expected limitations on smaller objects like barcodes (65% precision), consistent with known SSD characteristics [6].

5.2 Text Recognition Performance

Table 6. OCR performance comparison

Model	Word Acc. (%)	CER (%)	Latency (ms)	Size (MB)
Tesseract 4.0	67.2	18.4	340	85.0
PaddleOCR	92.5	3.8	210	8.6
CRNN (FP32)	93.2	3.2	184	28.4
DrishT CRNN (INT8)	**91.5**	**4.1**	**150**	**6.8**

Table 7. OCR performance by text type

Text Type	Samples	Word Acc. (%)	CER (%)
Printed (clear)	1,200	94.3	2.8
Printed (low light)	600	87.1	6.2
Handwritten	450	82.4	9.1
Signage	750	91.8	4.3
Overall	**3,000**	**91.5**	**4.1**

5.3 Semantic Fusion Performance

Table 8. Object-text association accuracy (Model Awareness Score)

Method	MAS (%)	Precision	Recall	F1-Score
Independent pipelines	71.6	0.74	0.69	0.71
Distance-only	76.2	0.79	0.72	0.75
IoU-only	78.4	0.81	0.75	0.78

Method	MAS (%)	Precision	Recall	F1-Score
DrishT (Combined)	**84.3**	**0.87**	**0.81**	**0.84**

The combined fusion approach achieves **12.7% absolute improvement** over independent pipelines, demonstrating the value of multi-metric spatial reasoning.

5.4 End-to-End System Performance

Table 9. Latency breakdown on Snapdragon 660

Component	Latency (ms)	% Total
Image preprocessing	8	4.4%
SSD inference	32	17.6%
CRNN inference	150	82.4%
Fusion layer	3	1.6%
Post-processing	9	4.9%
Total	**182**	**100%**

System Metrics: - Throughput: 5.5 FPS - Combined model size: 14.5 MB - Peak RAM usage: ~350 MB - Target device cost: <$200 (Snapdragon 660+ devices)

5.5 Ablation Studies

5.5.1 Quantization Impact

Table 10. Effect of INT8 quantization

Configuration	mAP (%)	OCR Acc. (%)	Latency (ms)	Size (MB)
Both FP32	43.0	93.2	421	89.4
SSD INT8 only	41.2	93.2	366	42.9
CRNN INT8 only	43.0	91.5	387	67.8
Both INT8	**41.2**	**91.5**	**332**	**21.3**

Key Finding: INT8 quantization achieves 76% size reduction and 21% latency reduction with only 1.8% mAP and 1.7% OCR accuracy degradation—an acceptable tradeoff for mobile deployment.

5.5.2 Fusion Component Ablation

Table 11. Contribution of fusion components

Configuration	MAS (%)	Δ vs Baseline

Configuration	MAS (%)	Δ vs Baseline
No fusion (baseline)	71.6	—
+ Distance weighting	76.2	+4.6
+ IoU scoring	80.1	+8.5
+ Vertical alignment	82.7	+11.1
+ Priority weighting	**84.3**	**+12.7**

Each component contributes statistically significant improvement ($p<0.05$, McNemar's test).

5.6 Real-World Evaluation

Evaluation on 1,000 real-world navigation images demonstrates practical utility:

Table 12. Real-world scenario performance

Scenario	Images	Success Rate	Primary Challenge
Menu/Product scanning	250	92%	Small text, glare
Room number reading	200	88%	Variable fonts
Exit sign detection	150	94%	Lighting variation
License plate reading	200	78%	Distance, angle
Stair detection	200	86%	Partial visibility

6. Discussion

6.1 Key Findings

1. **Quantization viability:** 1.8% mAP loss for 76% size reduction demonstrates the feasibility of INT8 quantization for mobile assistive technology without significant user-facing quality degradation.

2. **Semantic fusion value:** The 12.7% improvement in object-text association validates our hypothesis that integrated spatial reasoning outperforms independent pipeline approaches.

3. **Mobile deployment feasibility:** Achieving 182ms latency on Snapdragon 660 devices (available for <$200) demonstrates that sophisticated assistive AI can be accessible to economically disadvantaged populations.

6.2 Limitations

6.2.1 Technical Limitations

Small Object Detection: AP for small objects (barcodes, distant signs) remains at 65%, compared to 85% for large objects. This is a fundamental limitation of the SSD architecture at 300×300 input resolution.

Extreme Lighting: Performance degrades significantly in very dark (<5 lux) or high-contrast backlit conditions. Integration with device flashlight control could partially address this.

Crowded Scenes: Fusion accuracy drops to 72.3% when more than 5 objects are detected simultaneously, due to increased ambiguity in spatial association.

Language Support: Current implementation supports English alphanumeric characters only. Extension to other scripts requires expanded character sets and potentially different architecture choices.

6.2.2 Evaluation Limitations

Dataset Bias: Training data primarily represents indoor university and commercial environments. Performance in outdoor urban settings, particularly with weather variations, requires further validation.

User Study Scope: While we evaluated on navigation-relevant scenarios, formal user studies with visually impaired participants were not conducted in this initial work. Such studies are essential for validating real-world utility and should be a priority for future work.

6.3 Ethical Considerations

Accessibility: By enabling deployment on devices under $200 (vs. $3,500 for OrCam MyEye), DrishT addresses the economic barrier to assistive technology access.

Privacy: All processing occurs on-device with no cloud transmission, ensuring user privacy. However, the camera-based nature of the system raises awareness considerations that should be addressed in deployment guidelines.

Safety: The system is intended to augment, not replace, traditional mobility aids (white cane, guide dog). Clear disclaimers about system limitations are essential.

6.4 Comparison with Commercial Systems

Table 13. Comparison with existing solutions

Feature	OrCam	Seeing AI	Lookout	DrishT
Offline operation	✓	✗	Partial	✓
Real-time (<200ms)	✓	✗	✓	✓
Semantic fusion	Limited	✗	Limited	✓

Feature	OrCam	Seeing AI	Lookout	DrishT
Device cost	$3,500	$200-1000	$200-1000	<$200
Open source	✗	✗	✗	✓

7. Future Work

7.1 Short-Term Improvements (3–6 months)

1. **Higher resolution processing:** Increase input resolution to 512×512 for improved small object detection, with corresponding latency optimization.
2. **Multi-language OCR:** Extend character set to support Hindi, Chinese, and Arabic scripts, critical for global accessibility.
3. **Low-light enhancement:** Integrate preprocessing for automatic image enhancement in poor lighting conditions.

7.2 Medium-Term Research (6–12 months)

1. **Unified multi-task architecture:** Replace dual-stream pipeline with a shared backbone approach (e.g., DETR-based) to reduce redundant computation.
2. **Temporal consistency:** Leverage video-based tracking to smooth detections and reduce false positives across frames.
3. **Formal user studies:** Conduct IRB-approved studies with visually impaired participants to validate real-world utility and gather feedback for iterative improvement.

7.3 Long-Term Vision (1–2 years)

1. **Scene graph generation:** Move beyond bounding boxes to semantic scene understanding with object relationships.
2. **Natural language interaction:** Enable conversational queries (“Where is the nearest exit?”).
3. **Collaborative improvement:** Federated learning from user corrections while preserving privacy.

8. Conclusion

We presented DrishT, a mobile-optimized dual-stream architecture achieving state-of-the-art performance for assistive navigation through semantic fusion of object detection and scene text recognition. Our key contributions include:

1. **Semantic Fusion Architecture:** Distance-weighted attention fusion achieving 84.3% object-text association accuracy, representing a 12.7% improvement over baseline approaches.

2. **Mobile Optimization:** INT8 quantization reducing model size by 76% (14.5 MB total) and latency by 23% (182ms) with minimal accuracy degradation (<2%).

3. **Accessibility Impact:** Enabling deployment on devices costing under $200, reducing the economic barrier to assistive technology by 96% compared to proprietary alternatives.

While limitations remain in small object detection and extreme lighting conditions, DrishT demonstrates that sophisticated AI-powered assistive technology can operate on affordable consumer devices without cloud dependency. Future work will focus on unified multi-task architectures, extended language support, and formal user studies to validate real-world utility.

By documenting our methodology, results, and limitations transparently, we aim to provide a foundation for continued research in accessible AI systems.

Acknowledgments

The author thanks Dr. Narayan Kulkarni for supervision and guidance throughout this project. This work was completed as part of the Bachelor of Science degree requirements at Amity Institute of Information Technology, Amity University Maharashtra. Computational resources were provided by Google Cloud AI Platform.

Data Availability

The preprocessing pipeline and model configuration code are available upon reasonable request. Due to licensing restrictions on base datasets (COCO, ICDAR), we cannot redistribute training data directly but provide detailed instructions for dataset preparation.

References

[1] World Health Organization, “World Report on Vision,” 2019. [Online]. Available: https://www.who.int/publications/i/item/9789241516570

[2] R. L. Manduchi and S. Kurniawan, “Mobility-related accidents experienced by people with visual impairment,” *Research and Practice in Visual Impairment and Blindness*, vol. 4, no. 2, pp. 44–54, 2011.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proc. IEEE CVPR*, 2014, pp. 580–587.

[4] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *Proc. NeurIPS*, 2015, pp. 91–99.

[5] J. Redmon and A. Farhadi, “YOLOv3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single shot multibox detector,” in *Proc. ECCV*, 2016, pp. 21–37.

[7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “MobileNets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.

[8] M. Tan, R. Pang, and Q. V. Le, “EfficientDet: Scalable and efficient object detection,” in *Proc. IEEE CVPR*, 2020, pp. 10781–10790.

[9] R. Smith, “An overview of the Tesseract OCR engine,” in *Proc. ICDAR*, 2007, pp. 629–633.

[10] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, “EAST: An efficient and accurate scene text detector,” in *Proc. IEEE CVPR*, 2017, pp. 5551–5560.

[11] B. Shi, X. Bai, and C. Yao, “An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 11, pp. 2298–2304, 2017.

[12] H. Li, P. Wang, C. Shen, and G. Zhang, “Show, attend and read: A simple and strong baseline for irregular text recognition,” in *Proc. AAAI*, 2019, pp. 8610–8617.

[13] PaddlePaddle, “PaddleOCR: Awesome multilingual OCR toolkits,” GitHub, 2020. [Online]. Available: https://github.com/PaddlePaddle/PaddleOCR

[14] D. Ahmetovic, C. Gleason, C. Rez, K. Kitani, H. Takagi, and C. Asakawa, “NavCog: A navigational cognitive assistant for the blind,” in *Proc. MobileHCI*, 2016, pp. 90–99.

[15] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks,” in *Proc. ICML*, 2006, pp. 369–376.

[16] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common objects in context,” in *Proc. ECCV*, 2014, pp. 740–755.

[17] Voxel51, “FiftyOne: The open-source tool for building high-quality datasets and computer vision models,” 2020. [Online]. Available: https://voxel51.com/fiftyone/

[18] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny, “ICDAR 2015 competition on robust reading,” in *Proc. ICDAR*, 2015, pp. 1156–1160.

[19] A. Gupta, A. Vedaldi, and A. Zisserman, “Synthetic data for text localisation in natural images,” in *Proc. IEEE CVPR*, 2016, pp. 2315–2324.

Appendix A: Network Architecture Details

A.1 SSD-VGG16 Layer Configuration

Input: 300×300×3 RGB

VGG16 Base Network:
├── conv1_1: 64@3×3, ReLU → conv1_2: 64@3×3, ReLU → pool1: 2×2
├── conv2_1: 128@3×3, ReLU → conv2_2: 128@3×3, ReLU → pool2: 2×2
├── conv3_1-3: 256@3×3×3, ReLU → pool3: 2×2
├── conv4_1-3: 512@3×3×3, ReLU → pool4: 2×2 [Detection head 1]
└── conv5_1-3: 512@3×3×3, ReLU → pool5: 3×3, stride=1

Extra Layers:
├── fc6: 1024@3×3 (dilated), ReLU, Dropout(0.5)
├── fc7: 1024@1×1, ReLU, Dropout(0.5) [Detection head 2]
├── conv8_1: 256@1×1 → conv8_2: 512@3×3, stride=2 [Detection head 3]
├── conv9_1: 128@1×1 → conv9_2: 256@3×3, stride=2 [Detection head 4]
├── conv10_1: 128@1×1 → conv10_2: 256@3×3 [Detection head 5]
└── conv11_1: 128@1×1 → conv11_2: 256@3×3 [Detection head 6]

Detection Heads (6 scales):
├── Feature map sizes: {38×38, 19×19, 10×10, 5×5, 3×3, 1×1}
├── Anchors per location: {4, 6, 6, 6, 4, 4}
└── Total anchors: 8,732

A.2 CRNN Layer Configuration

Input: 32×W×3 (variable width)

CNN Feature Extractor:
├── conv1: 64@3×3, BN, ReLU, MaxPool(2×2, stride=2) → 16×W/2×64
├── conv2: 128@3×3, BN, ReLU, MaxPool(2×2, stride=2) → 8×W/4×128
├── conv3_1: 256@3×3, BN, ReLU
├── conv3_2: 256@3×3, BN, ReLU, MaxPool(2×1, stride=2×1) → 4×W/4×256
├── conv4_1: 512@3×3, BN, ReLU
├── conv4_2: 512@3×3, BN, ReLU, MaxPool(2×1, stride=2×1) → 2×W/4×512
└── conv5: 512@3×3, BN, ReLU → 1×W/4×512

Sequence Modeling:
├── Reshape: (W/4)×512
├── BiLSTM-1: 256 hidden → (W/4)×512
└── BiLSTM-2: 256 hidden → (W/4)×512

Transcription:
├── FC: 512 → 63 (62 chars + blank)
└── CTC Decode: beam_width=10

Appendix B: Hyperparameter Summary

```yaml
# SSD Training Configuration
ssd:
  optimizer: SGD
  learning_rate:
    initial: 0.004
    schedule: cosine_annealing
    final: 0.0001
  momentum: 0.9
  weight_decay: 0.0005
  batch_size: 32
  epochs: 100
  warmup_epochs: 5
  input_size: [300, 300]
  neg_pos_ratio: 3

# CRNN Training Configuration
crnn:
  optimizer: Adam
  learning_rate:
    initial: 0.001
    schedule: reduce_on_plateau
    factor: 0.5
    patience: 5
  beta1: 0.9
  beta2: 0.999
  batch_size: 32
  epochs: 50
  input_height: 32
  max_text_length: 25

# Quantization Configuration
quantization:
  method: post_training_int8
  calibration_samples: 1000
  per_channel_conv: true
  per_tensor_lstm: true
```