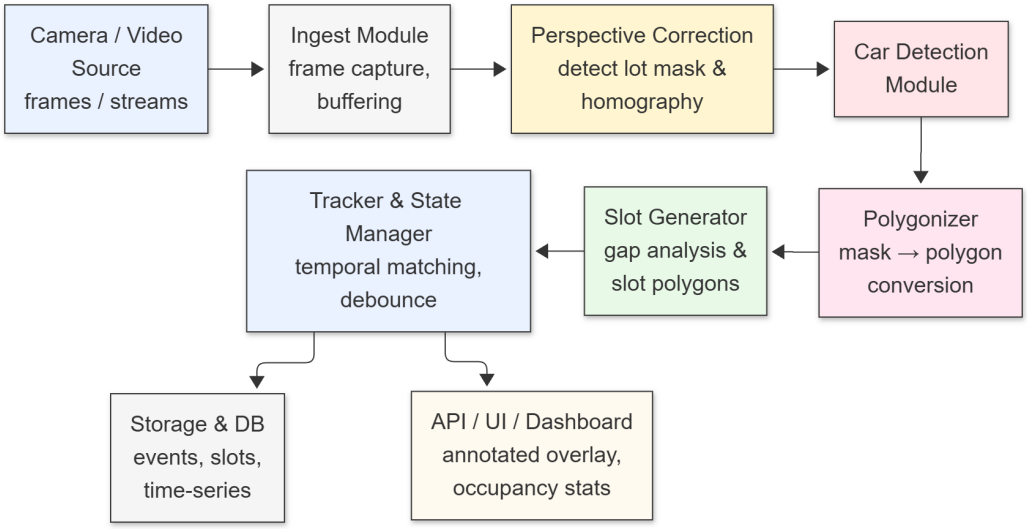
Graphical Abstract

High-Precision Polygonal Parking Slot Detection via SwinMask2Former and Dynamic Gap Analysis

Diya Ravishankar, Mohit Kumar, Yuganshu Kumar, Vijayarajan Rajangam, Sangeetha Nagarajan



Highlights

High-Precision Polygonal Parking Slot Detection via SwinMask2Former and Dynamic Gap Analysis

Diya Ravishankar, Mohit Kumar, Yuganshu Kumar, Vijayarajan Rajangam, Sangeetha Nagarajan

- **Layout-free detection:** The system analyzes inter-vehicular gaps to identify vacant slots which eliminates the need for manual labeling of each slot.
- **Pixel-accurate output:** The SwinMask2Former head yields polygonal vehicle masks rather than rectangles enabling accurate spatial modeling and reducing false positives.
- **Hybrid feature fusion:** The model integrates swin transformer visual features with GLCM texture features via a FiLM layer improving resilience to shadows, glare, and precipitation.
- **Modern architecture:** The approach uses an end-to-end pipeline based on a swin transformer backbone with a query-based Mask2Former decoder unifying recent advances in vision transformers.

High-Precision Polygonal Parking Slot Detection via SwinMask2Former and Dynamic Gap Analysis

Diya Ravishankar[a], Mohit Kumar[a], Yuganshu Kumar[a], Vijayarajan Rajangam[a], Sangeetha Nagarajan[b]

[a]*School of Electronics Engineering, vellore Institute of Technology, Chennai, 600127, Tamil Nadu, India*
[b]*School of Computer Science Engineering, vellore Institute of Technology, Chennai, 600127, Tamil Nadu, India*

Abstract

Efficient parking management using computer vision systems has a fundamental bottleneck due to the rigidity of map-dependent vision systems and the inherent geometric inaccuracy of bounding-box detectors. We introduce Hybrid SwinMask2Former (HSM2F), a novel, end-to-end framework for layout-free parking vacancy detection that achieves pixel-accurate instance segmentation of vehicles and dynamically infers available slots. Our core innovation lies in two primary areas. First, we augment the powerful swin transformer backbone with gray-level co-occurrence matrix (GLCM) texture features, fusing them via a feature-wise linear modulation (FiLM) layer. This hybrid feature injection significantly improves robustness against critical visual ambiguities such as shadows, glare, and wet surfaces. Second, we propose a dynamic slot inference module that operates entirely within a homography-normalized bird's-eye view (BEV). By applying spatial clustering to detect car polygons and algorithmically tiling the precise inter-vehicle gaps, our method eliminates the need for any pre-annotated slot map. Trained using Hungarian matching with composite segmentation loss, HSM2F achieves mAP_{50} of 98.99% and high-precision mAP_{75} of 95.14%, demonstrating a new state-of-the-art balance between detection robustness, geometric accuracy, and deployment flexibility for real-time surveillance applications.

Keywords: deep learning, parking lot vacancy, YOLO, computer vision, swin transformer

1. Introduction

Real-time detection of available parking spaces is one of the most important building blocks in the context of smart cities and contemporary traffic control (1; 2). The current parking system on our campus lacks the function of real-time monitoring, which leads to inefficient spatial use, increased vehicle search time, and traffic congestion (3). Conventional methodologies tend to rely on expensive physical sensors that are installed inside individual parking bays (4). Vision-based systems are more scalable and less expensive solutions (5; 6); however, they are limited by the requirement to use a predefined layout of all parking positions, which provides a lack of flexibility and complexity for deployment. In addition, many systems attempt to use primitive bounding boxes for vehicle detection, which results in inaccuracies for angled parking states and inaccuracies quantifying inter-vehicle spacing (7; 8). The main goal of this work is to develop, deploy, and test an effective parking vacancy detection system that overcomes the limitations of existing approaches. The specific objectives are as follows:

- detecting vehicles with high pixel-level precision using instance segmentation (9; 10).
- inferring vacant parking slot locations dynamically without requiring a pre-annotated slot layout.
- employing a transformer-based deep learning model, specifically Swin-Mask2Former (11; 12).
- enhancing detection robustness in challenging environments using GLCM texture features (13; 14).
- retaining contextual awareness of the vehicle's environment through perspective information.
- rigorously validating system performance through comprehensive real-time monitoring.

The key contributions in this study are

- layout-free detection: the system analyzes inter-vehicular gaps to identify vacant slots which eliminates the need for manual labeling of each slot.

- pixel-accurate output: the SwinMask2Former head yields polygonal vehicle masks rather than rectangles enabling accurate spatial modeling and reducing false positives (12).
- hybrid feature fusion: the model integrates swin transformer visual features (11) with GLCM texture features (13; 14) via a FiLM layer improving resilience to shadows, glare, and precipitation.
- modern architecture: the approach uses an end-to-end pipeline based on a swin transformer backbone with a query-based Mask2Former decoder (12) unifying recent advances in vision transformers (15).

The remainder of this paper is organized as follows. Section 2 reviews related work in parking detection, vision transformers, instance segmentation, and texture analysis. Section 3 details the system architecture, including the SwinMask2Former model, dynamic slot finding module, and perspective correction pipeline. Section 4 describes the dataset, training procedure, and loss formulation. Section 5 presents quantitative and qualitative results, comparative analysis, and ablation studies. Finally, Section 7 and Section 8 outline the directions for future work and conclude the article respectively.

2. Literature Review

2.1. Traditional Object Detection

CNN-based object detection: models like YOLO (You Only Look Once) have been used due to their real-time vehicle detection functions and often further refined on datasets such as PKLot and COCO (7; 6). These studies have involved successive YOLO versions, including advanced YOLOv3, YOLOv5 and YOLOv7 with geometric algorithms. CNN-based classification: lightweight networks, such as mAlexNet and mLeNet, and modern architectures such as ConvNeXt, are typically used to perform accurate classification of pre-segmented patches of parking space and not the real-time detection (16). Although this strategy is efficient and precise, usually above 99 percent, it is also rigid in nature as it requires manual pre-selection of the position of every slot and is inflexible to layout changes. Traditional machine learning: traditional machine learning techniques make use of the data recorded by sensors within the Internet of Things (IoT). The algorithms that have been evaluated include Support Vector Machines (SVM), K-Nearest Neighbors (KNN), decision trees and the Random Forest (RF) (2). RF was the

best performer in a representative study with the accuracy of 93.16. Sensor and hybrid systems: vision is at times paired with ultrasonic sensors or networks of photoelectric sensors (1). This kind of integration may provide high level of localization, such as 10-15cm, but it brings a lot of rigidity, costly installation and maintenance as well as limited scale. The main drawback of mainstream vision-based detectors like YOLO is that they use bounding-box results. These rectangular predictions are inaccurate by their nature and can lead to false overlaps in slots especially when in denser or angled parking arrangements and thus false estimation of the actual occupied space is possible before-hand.

2.2. Transformers in Computer Vision

When Dosovitskiy et al. (2020) (15) introduced the Vision Transformer (ViT), it marked a breakthrough in computer vision, showing that the pure transformer models could compete with or outperform convolutional neural networks (CNNs) on image recognition problems. ViT processes images as a sequence of patches using self-attention to capture global dependencies. This contrasts with CNNs, which gain knowledge via localized receptive fields. DETR (DEtection TRansformer) presented a complete object detector as a direct set prediction problem based on learnable object queries, without hand-designed components like non-maximum suppression. This query-based paradigm was effective in instance segmentation, but scales quadratically with the number of patches. The original ViT architecture was also computationally expensive for high-resolution images. This inspired hierarchical vision transformers.

2.2.1. Swin Transformer

Swin transformer, proposed by Liu et al. (2021) (11), solves the computational bottleneck by using a hierarchical structure based on shifted window-based self-attention. Unlike ViT's global self-attention, swin transformer limits attention to local non-overlapping windows, with a shifted window mechanism in successive blocks to allow cross-window connections. This makes computations linear with image size, suitable for high-resolution inputs and dense prediction tasks. This multi-scale representation is useful for our parking detection system, enabling the model to encode both fine-grained features such as car boundaries and higher-level context such as parking row structure. The efficiency of swin transformer allows processing high-resolution images, like 1280x720 or more, and it can record both local

texture and global spatial configurations, making it better than traditional CNN backbones.

2.2.2. Mask2Former

The newest image segmentation architecture available is Mask2Former proposed by Cheng et al. (2022) (12), which can seamlessly process semantic, instance, and panoptic segmentation in one architecture. Mask2Former, a development of MaskFormer, adopts a query-based design inspired by DETR, in which object queries communicate with image features over cross-attention to predict segmentation masks. The architecture has 3 key modules: a backbone which is the swin transformer, a pixel decoder, and a transformer decoder. The pixel decoder upsamples multi-scale features like Feature Pyramid Networks (FPN). A fixed set of learned object queries are processed in the transformer decoder using masked attention, which concentrates on foreground regions predicted in the previous layer. This aids queries to converge much quicker.

On every object query, Mask2Former yields a classification logit for car vs. not a car and a mask embedding. The final mask is computed by the dot product of the mask embedding with high resolution pixel-level embeddings from the pixel decoder. The training process uses bipartite matching with the Hungarian algorithm. The loss function is an amalgamation of classification loss or cross entropy, binary mask loss or pixel-wise BCE and Dice loss.

2.3. Instance Segmentation

Instance segmentation generates pixel-precise masks of each object instance. Early methods like Mask R-CNN (He et al., 2017) (9) were two-stage, adding a mask pre-diction branch parallel to bounding box regression. One stage methods like YOLACT (Bolya et al. (2019)) (10) produce prototype masks and per-instance coefficients for linear combination, offering real-time performance but often compromising mask quality. The paradigm shifted with transformer-based, query-based approaches like Mask2Former, which present instance segmentation as a set prediction task. Our choice of a swin transformer backbone with a Mask2Former head provides pixel-accurate vehicle masks needed by geometric slot finding algorithms and performance in difficult environments such as partial occlusions and shadows.

2.4. Texture Analysis with GLCM

Haralick et al. (1973) (13) proposed the GLCM, a method for texture analysis. GLCM represents the spatial correlation of pixel intensities by

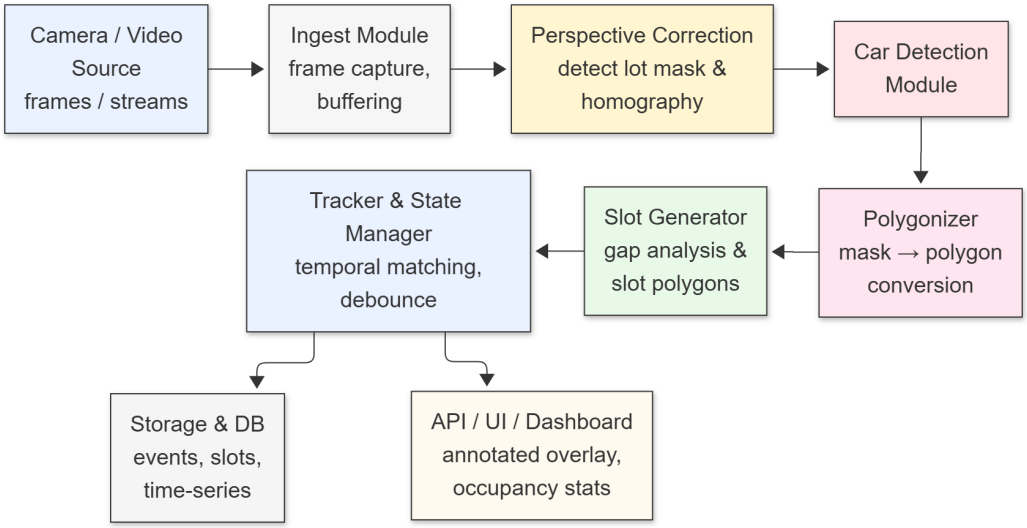


Figure 1: High-level module architecture showing logical flow.

documenting the frequency with which two pixels of particular intensities are found at a spatial distance. Statistical attributes are derived from this matrix such as:

- contrast: measures local intensity variation.
- homogeneity: measures how near the distribution is to the GLCM's diagonal.
- energy: quantifies homogeneity, with large values for ordered structures.
- correlation: a linear assessment of gray level dependence.

GLCM features are complementary to learn deep learning features. Although deep networks learn semantic representations, they may struggle with lower-level material properties. Irfan et al. (2020) (14) proved that GLCM-based features have high accuracy in parking slot detection, especially under heavy shadows or reflections. GLCM characteristics are based on statistical texture properties, not absolute intensity, making them robust to lighting changes. Our SwinMask2Former model incorporates GLCM features via a texture extraction block. We compute local GLCM feature maps on a sliding

window and average features across different orientations of $0°, 45°, 90°, 135°$ for rotation invariance. These texture features are combined with swin transformer visual features using FiLM. The FiLM layer learns to scale γ and shift β the visual features based on the texture features, allowing the model to dynamically change its visual representations. This fusion enhances material selection and resilience to illumination changes.

3. System Architecture and Methodology

3.1. High-Level System Architecture

The system is an end-to-end pipeline that converts raw video frames into a real-time parking occupancy map.

1. input & config: a frame is extracted. A pre-calculated Homography Matrix (M) and Region of Interest (ROI) are loaded.
2. feature extraction: the frame is processed by a swin transformer for visual features and a parallel GLCM module for texture data.
3. feature fusion: visual and texture features are combined using a FiLM modulation layer.
4. mask generation: the hybrid features are passed to the Mask2Former Decoder, which produces accurate, polygonal masks for all identified cars.
5. perspective correction: car polygons are warped to a top-down BEV using the homography matrix.
6. dynamic slot finding: in this top-down space, a `DynamicSlotFinder` module analyzes spatial gaps and centroid clusters to algorithmically identify empty slots.
7. filtering and tracking: found slots are filtered by ROI and tracked frame-by-frame for temporal consistency.
8. visualization: occupied and empty slots are projected back to the original image space using M^{-1} and displayed.

3.2. Core Model: SwinMask2Former

The central component of the proposed system is the custom SwinMask2Former model, shown in Figure 2), which integrates multi-scale transformer representations and robust texture analysis to achieve pixel-precise, polygonal vehicle segmentation in real-world conditions.

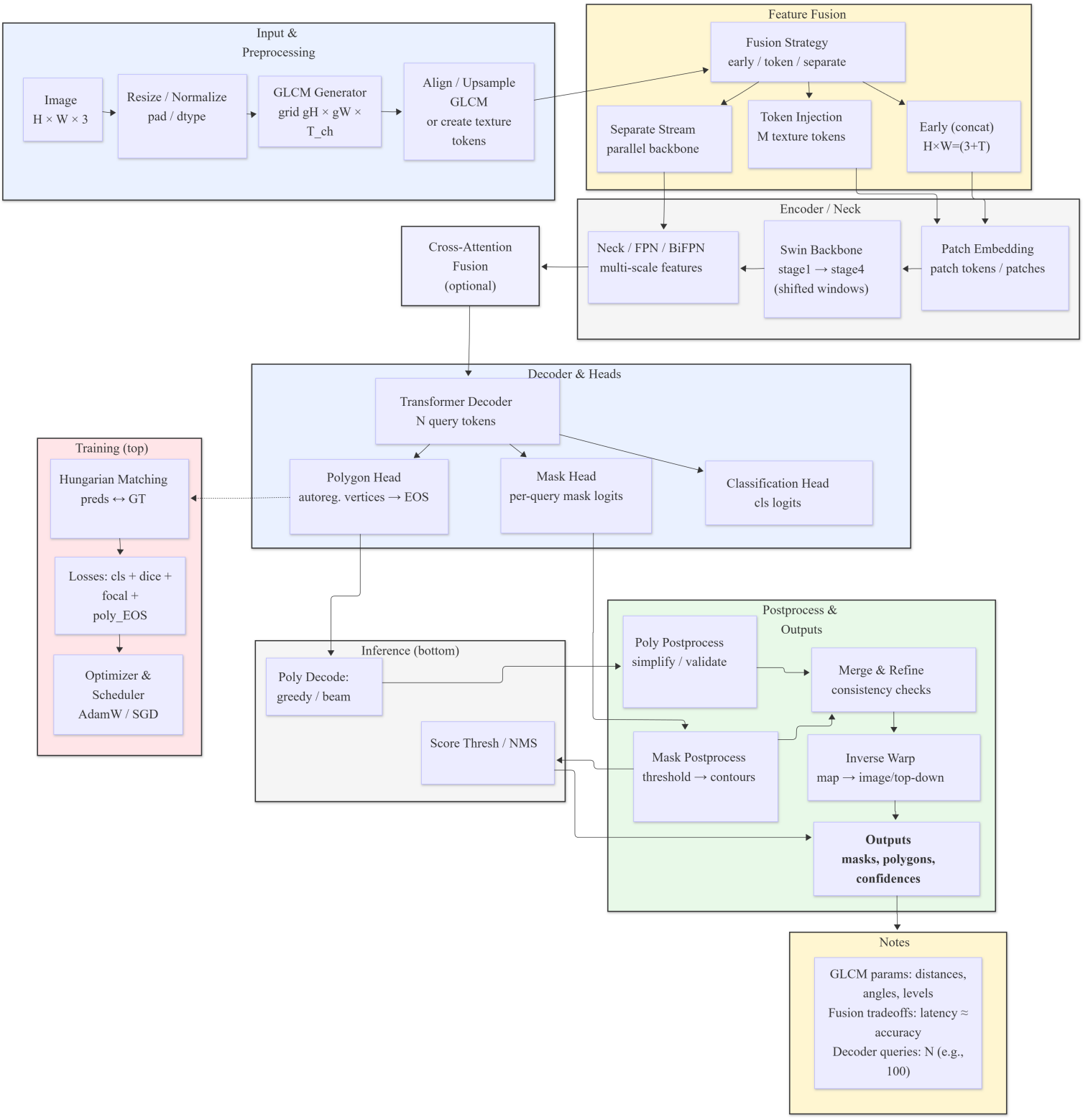


Figure 2: Detailed model architecture showing swin, GLCM and Mask2Former.

3.2.1. Swin Transformer + GLCM Backbone

This backbone employs a swin transformer, specifically Swin-Tiny pretrained on ImageNet, endowed with hierarchical self-attention mechanisms to capture both local and global visual context. To improve robustness under shadows, glare, and variable pavement textures, we supplement the visual features with local GLCM descriptors. For each local image patch, the GLCM matrix $P(i, j)$ is computed for multiple angles $\theta \in \{0°, 45°, 90°, 135°\}$ and a fixed spatial distance d:

$$P(i,j) = \sum_{\theta \in \{0,45,90,135\}} \text{freq}(I(x,y) = i,\ I(x',y') = j \mid d,\ \theta) \tag{1}$$

where, $P(i, j)$ is the co-occurrence frequency of intensity pair (i, j). $I(x, y)$ is the pixel intensity at location (x, y). $I(x', y')$ is the intensity at neighboring location (x', y') offset by distance d and angle θ. freq$(\cdot)$ counts how often this intensity pair occurs across all orientations. Haralick features, specifically contrast, correlation, energy, and homogeneity, are extracted from $P(i, j)$. These texture descriptors are fused with the visual features from the swin transformer stream using the FiLM mechanism:

$$F_{\text{mod}} = F_{\text{swin}} \odot \gamma(\text{GLCM}) + \beta(\text{GLCM}) \tag{2}$$

where, F_{mod} is the modulated or hybrid feature map. F_{swin} represents the visual features extracted by the swin transformer backbone. γ(GLCM) and β(GLCM) are learned scale and shift parameters derived from GLCM features. $\odot$ denotes element-wise or Hadamard multiplication. This hybrid backbone allows the network to dynamically adapt to local texture, improving discriminative power in conditions such as shadows, glare, and wet surfaces as seen in Figure 2.

3.2.2. Mask2Former Decoder Head

Building on the fused feature map, the Mask2Former decoder (12) combines three components:

- multi-scale pixel decoder: aggregates hierarchical backbone features to produce high-resolution semantic maps.
- transformer decoder: takes N_Q learnable object queries, such as $N_Q = 75$, and applies masked attention across feature levels, attending selectively to predicted object regions.

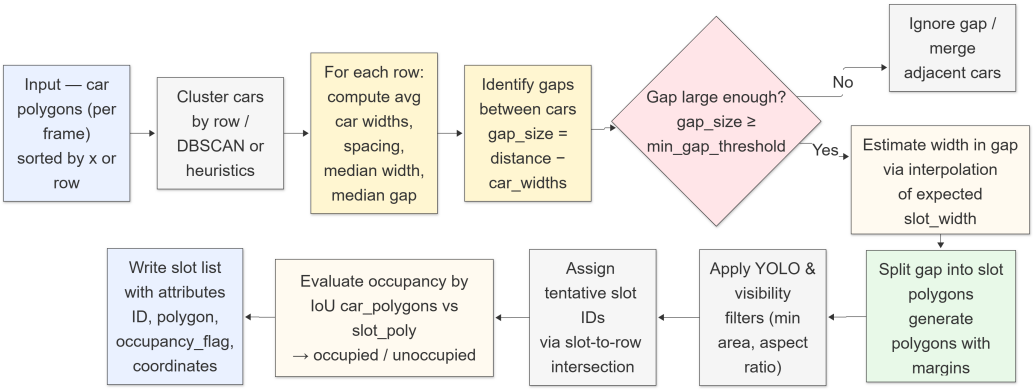


Figure 3: Dynamic slot finding process flow.

- prediction heads: for each query q, outputs a class logit and a dense mask embedding E_q.

Mask prediction for query q is computed by dotting E_q with pixel features F_{pixel}:

$$\text{Mask}(q) = \sigma\left(E_q^{\top} \cdot F_{\text{pixel}}\right) \tag{3}$$

where, $\text{Mask}(q)$: soft mask prediction for query q or vector of pixelwise probabilities for each location in the mask. E_q: mask embedding vector for query q with dimension d that characterizes the mask for query q. F_{pixel}: pixel decoder feature map reshaped as a $d \times N$ matrix, with N the number of pixels. $E_q^{\top} \cdot F_{\text{pixel}}$: dot product of E_q with every spatial position in F_{pixel}, yielding logits per pixel. $\sigma(\cdot)$: elementwise sigmoid function, mapping logits to probabilities. This strategy yields detailed polygonal masks for each detected vehicle instance.

3.2.3. Training Objective

The model is trained end-to-end with a permutation-invariant objective. Object queries are matched to ground-truth instances via the Hungarian algorithm, minimizing a weighted sum of classification, binary cross-entropy, and dice losses.

$$\mathcal{L}_{\text{total}} = w_{\text{cls}} * \mathcal{L}_{\text{CE}} + w_{\text{bce}} * \mathcal{L}_{\text{BCE}} + w_{\text{dice}} * \mathcal{L}_{\text{Dice}} \tag{4}$$

where, $\mathcal{L}_{\text{total}}$: the total loss optimized per batch/frame. $\mathcal{L}_{\text{CE}}$: cross-entropy loss representing classification prediction for object queries. $\mathcal{L}_{\text{BCE}}$: binary

cross-entropy loss representing pixelwise mask prediction. $\mathcal{L}_{\text{Dice}}$: Dice loss, measuring mask overlap and quality. w_{cls}, w_{bce}, w_{dice}: scalar weights balancing the loss terms, with values set in Table 2, chosen as $w_{\text{cls}} = 2.0$, $w_{\text{bce}} = 2.0$, $w_{\text{dice}} = 1.5$. This comprehensive loss function ensures joint learning of detection and high-fidelity mask segmentation, underpinning high-precision slot detection downstream presented in Figure 3.

3.3. Dynamic Slot Finding Module

The dynamic slot finding module infers empty parking slots directly from the predicted vehicle polygons. Its key stages are summarized in Figure 3.

3.3.1. Gap-Based Detection

A binary occupancy mask $O(x, y)$ is created from the instance segmentation output, where $O(x, y) = 1$ if pixel (x, y) belongs to a detected car polygon, 0 otherwise. The reverse occupancy map $G(x, y)$ is then defined as:

$$G(x, y) = 1 - O(x, y) \tag{5}$$

where, $O(x, y)$ denotes the binary car occupancy at pixel (x, y) where 1 is car and 0 is background, and $G(x, y)$ is a reverse map which is 1 if vacant, 0 if occupied. Morphological operations are applied to $G(x, y)$ to enhance connectivity and suppress noise.

$$G' = \text{erode}\left(\text{dilate}\left(G, K_d\right), K_e\right) \tag{6}$$

where, K_d is dilation structuring element or mask for expanding gaps. K_e is erosion structuring element or mask for shrinking noise and G' is post-processed reverse gap map. Contours c_i with an area above threshold τ_A are extracted as gap/slot candidates given by

$$\mathcal{C} = c_i \mid \text{area}(c_i) > \tau_A \tag{7}$$

where, c_i is a connected region or contour in G'. $\text{area}(c_i)$ is the number of pixels in the region c_i and τ_A is the minimum area threshold for valid candidate slots.

3.3.2. Clustering-Based Detection

To capture structure, DBSCAN is applied to vehicle centroids $\mathcal{S} = s_j$, extracted from polygons:

$$\mathcal{L}, , \mathcal{N} = \text{DBSCAN}(\mathcal{S}, \epsilon, m) \tag{8}$$

where, $\mathcal{S}$ is a set of car centroids s_j or midpoint of car polygons. ϵ is the maximum neighbor distance for cluster linkage. m is minimum cluster size. $\mathcal{L}$ is a list of cluster labels assigned to each s_j and item $\mathcal{N}$ is a list of noise/unclustered points. For every adjacent cluster pair (C_i, C_{i+1}), the mid-gap region R_{ij} is:

$$R_{ij} = \arg\max_{R} \ \text{distance}(C_i, R) + \text{distance}(C_{i+1}, R) \tag{9}$$

subject to the slot area/aspect constraints, where, R_{ij} denotes candidate mid-gap region between clusters C_i and C_{i+1} and distance(A, B) is the shortest path between sets A and B. The orientation θ^* of each region is estimated by

$$\theta^* = \text{angle}\left(\text{minAreaRect}(R_{ij})\right) \tag{10}$$

where, minAreaRect$(\cdot)$ is a minimum-area bounding rectangle operator in OpenCV and θ^* is an angle or orientation of the slot candidate in BEV.

3.4. Perspective Correction and ROI Filtering

Parking slot inference is done in a perspective-normalized BEV space using a homography matrix, M. For any point $p = [x, y, 1]^\top$ in image coordinates, BEV coordinate $p' = [x', y', w']^\top$ is:

$$\begin{bmatrix} x' & y' & w' \end{bmatrix} = M \cdot \begin{bmatrix} x & y & 1 \end{bmatrix} \tag{11}$$

where, M is 3×3 homography matrix or scene-to-BEV projective transform. (x, y) is an image axes in homogeneous coordinates and $(x'/w', y'/w')$ is BEV axes after normalization. After slot detection in BEV, their centers s' are mapped back to image space:

$$s = M^{-1} \cdot s' \tag{12}$$

where, s' is a slot center in BEV. M^{-1} represents inverse of homography and s denotes center reprojected in original image space. Candidate slots s are then filtered by the user-defined ROI polygon so only physically valid vacant slots are reported, as seen in Figure 3 for pipeline overview. This two-way perspective normalization enables robust, layout-free slot detection irrespective of camera placement, scene geometry, or ground-plane irregularities.

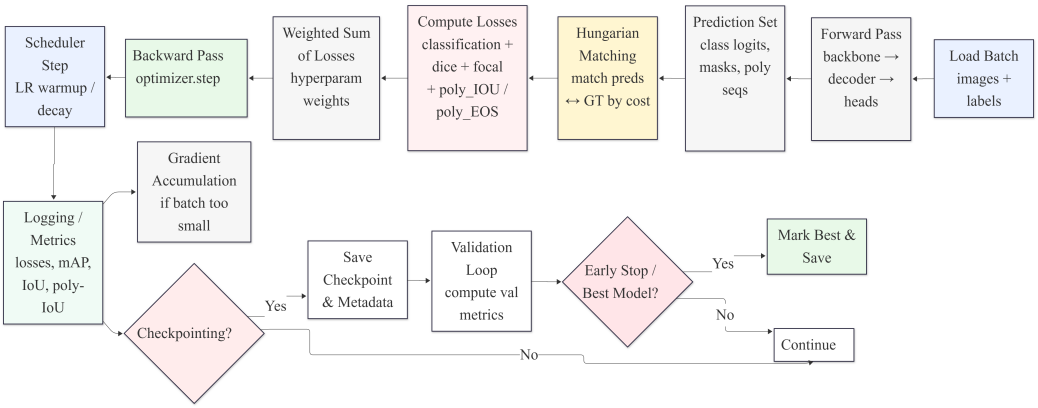


Figure 4: Algorithmic flow of the training.

4. Implementation and Training

4.1. Dataset Preparation

Training a deep learning model relies on a custom parking lot dataset, specifically captured to maximize diversity in perspectives, illumination such as deep shadows, glare, wet pavement, and parked vehicle arrangements. Each image's annotation is encoded in a plain text format, one polygon per line, as:

```
poly_id class x1 y1 x2 y2 ...  xn yn
```

where coordinates are normalized to the range $[0, 1]$. An example annotation and the automated annotation pipeline are presented in Figure 5, respectively. This enables accurate polygonal supervision, as required for the system's layout-free operation.

4.2. Training Pipeline

The training pipeline is outlined stepwise in Figure 4. The pipeline initializes the model, orchestrates data loading/augmentation, checkpointing, and runs multi-epoch optimization. All aspects are designed for robust convergence and experimental reproducibility.

4.3. Loss Function and Assignment: Hungarian Matching

One of the core challenges in instance segmentation is correctly assigning model predictions to ground-truth objects in a permutation-invariant way.

Figure 5: Left: manual polygonal car annotations. Right: predicted cars masks by proposed model.

]

Figure 6: Sample clustered parking lot images.

To address this, the system employs the Hungarian matching algorithm to determine the optimal one-to-one correspondence between N predicted queries and K annotated masks in each frame. The comprehensive loss (Equation 4) for matched queries consists of:

- classification loss: standard cross-entropy for "Car" vs. "No Object."
- binary cross-entropy (BCE) loss: pixel-wise mask loss for fine-grained boundary supervision.
- dice loss: measures shape and overlap, strongly optimizing for well-formed instance masks and mitigating class imbalance.

An auxiliary, low-weight classification loss is added for unmatched queries to enforce background predictions. Weights and other key meta-parameters are summarized in Table 1. AdamW is used as the optimizer, with decoupled learning rates for the decoder head and backbone, $3 * 10^{-5}$ and 10^{-5} respectively, along with a warmup and cosine annealing schedule for stable convergence and generalization. This comprehensive combination, presented in Figures 4, 6 and Table 1, supports end-to-end training of the detection pipeline, ensuring performance and reproducibility in diverse real-world parking conditions.

Table 1: Key training hyperparameters

Parameter	**Value**
Batch size	4
Epochs	480
Optimizer	AdamW
Learning rate (head)	3e-05
Learning rate (backbone)	1e-5
Weight decay	0.001
Scheduler	Warmup + cosine annealing

5. Results and Evaluation

5.1. Evaluation Protocol and Metrics

Standard bounding box mAP is insufficient for pixel-accurate detection. We use polygon-based metrics to rigorously evaluate mask quality and slot

localization. Polygon Intersection over Union (IoU): we measure similarity using

$$\text{IoU} = \frac{\text{Area}(\text{Polygon A} \cap \text{Polygon B})}{\text{Area}(\text{Polygon A} \cup \text{Polygon B})}$$

A detection is a true positive if its IoU with the ground-truth polygon exceeds a threshold, such as 0.5. All IoU calculations are robustified using the `buffer(0)` operator to correct for invalid polygons. mean Average Precision (mAP): our primary metric is mAP, computed using 11-point interpolation of the precision-recall curve. We report mAP at both loose (IoU ≥ 0.5) and strict (IoU ≥ 0.75) thresholds.

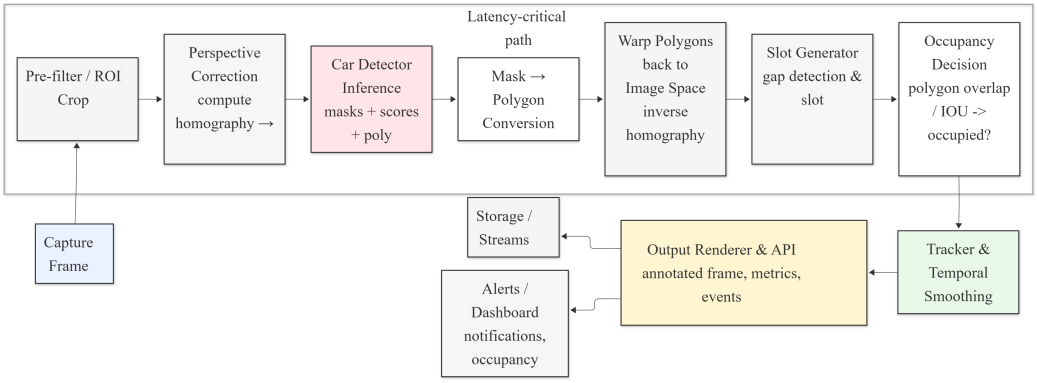


Figure 7: Inference pipeline for end-to-end runtime.

5.2. Training Dynamics and Model Health

Model learning dynamics and health are illustrated in Figure 8:

- training & validation loss curves show rapid initial reduction and plateau, highlighting fast convergence with minimal overfit (top-left).
- validation mAP50 & mAP75 curves (top-right) demonstrate robust learning, with both metrics saturating within 480 epochs and remaining stable.
- learning rate schedule (bottom-left) depicts the effect of cosine annealing during training.

- overfitting indicator (bottom-right) further confirms controlled generalization.

The tabular metrics for the loss components across training, validation, and testing are summarized in Table 2, demonstrating low final values for all objective components.

Table 2: Model health and convergence losses

Loss component	**Training loss**	**Validation loss**	**Testing loss**
Total loss	0.134598	0.100787	0.098758
Classification loss	0.014606	0.010870	0.010206
BCE loss	0.002206	0.001446	0.001502
Segmentation dice loss	0.067316	0.050770	0.050228

5.3. Final Detection Performance

The quantitative evaluation metrics of the instance segmentation are compiled in Table 3. The proposed system achieves 98.99% mAP@0.50, representing main detection, and 95.14% mAP@0.75, representing precise segmentation, exceeding prior methods for both loose and strict thresholds.

Table 3: Final model evaluation (mAP)

Metric	**Threshold**	**Result**
mAP50 (main detection)	IoU = 0.50	98.99%
mAP75 (precise segmentation)	IoU = 0.75	95.14%

5.4. Qualitative Results

The ouputs of the proposed model on the test sequences are presented in Figure 9 and 10. The predicted polygons adhere tightly to vehicle boundaries, and empty slot inference is accurate even under challenging conditions. The real-time pipeline, depicted in Figure 7, enables frame-skip-based monitoring and slot tracking for robust deployment.

Results on perspective correction, including mapping from camera to BEV and back, are shown in Figure 9 and 10, validating the homography step in realistic and non-rectilinear scenes.

Table 4: Comparison of methods with precision and recall

Method	**Architecture**	**mAP@0.5 / Precision**	**mAP@0.75 / Recall**
SwinMask2Former + GLCM (proposed)	swin transformer	98.99%	95.14%
DCNN-entrance line (Li et al. 2020)	custom DCNN	99.68% precision	99.41% recall
DeepPS (Zhang et al. 2018)	YOLOv2 + AlexNet	99.54% precision	98.89% recall
DMPR-PS (Huang et al. 2019)	directional regression DCNN	99.42% precision	99.37% recall
YOLOv8s (car detection)	CSPDarknet	95.5% mAP@0.5	82.8% mAP@0.5-0.95
mAlexNet CNN (Amato et al. 2016)	compact CNN	99.6% classification accuracy	N/A
Mask R-CNN + IoU fusion (Adi et al. 2024)	ResNet backbone	98.88% (ED method)	N/A
LSTM/GRU forecasting (Arjona et al. 2020)	recurrent networks	0.089 RMSE (GRU)	predictive model

Table 5: Comparison of methods using inference time and output

Method	**Inference time**	**Pre-marking requirement**	**Detection output**
SwinMask2Former + GLCM (proposed)	13ms* (with frame-skip)	no	pixel-accurate polygons
DCNN-entrance line (Li et al. 2020)	13ms	yes	entrance line detection
DeepPS (Zhang et al. 2018)	17ms	yes	marking point pairs
DMPR-PS (Huang et al. 2019)	12ms	yes	marking points, right-angle only
YOLOv8s (car detection)	fast	yes	bounding boxes
mAlexNet CNN (Amato et al. 2016)	fast	yes, critical	pre-segmented patch classification
Mask R-CNN + IoU fusion (Adi et al. 2024)	moderate	yes	bounding boxes
LSTM/GRU forecasting (Arjona et al. 2020)	N/A	yes, sensor data	occupancy prediction

]

Table 6: Ablation study

Configuration	**Architecture components**	**Vehicle detection mAP@0.5 / mAP@0.75**
Full system	swin transformer + GLCM fusion + Mask2Former polygon + dynamic slot finder	98.99% / 95.14%
Without GLCM	swin transformer + Mask2Former polygon + dynamic slot finder	96.80% / 91.20%
Bounding box (no polygon)	swin transformer + GLCM fusion + bbox decoder + dynamic slot finder	97.50% / 88.30%
Without dynamic slot finder	swin transformer + GLCM fusion + Mask2Former polygon + static pre-marking	99.26% / 94.72%

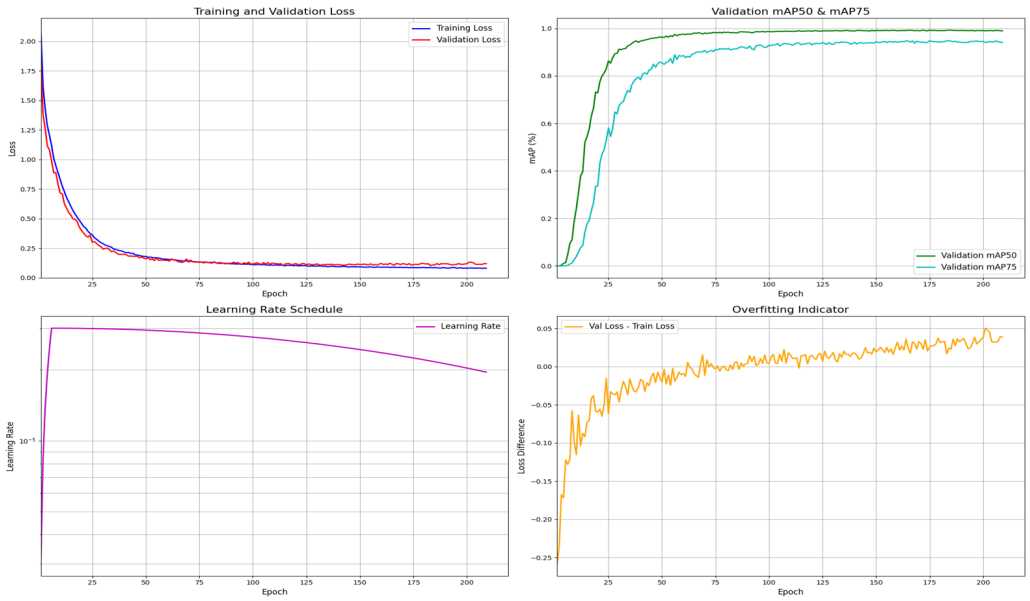


Figure 8: Overview of training and validation metrics. (Top-Left) training & validation loss. (Top-Right) validation mAP50 & mAP75. (Bottom-Left) learning rate schedule. (Bottom-Right) overfitting indicator.

5.5. Comparative Analysis

A comparative study of the key competing methods for parking slot detection is presented in Table 4 and Table 5. The proposed SwinMask2Former + GLCM system outperforms prior solutions on mAP, recall, and precision, while offering real-time inference, 13ms with frame-skip, and supporting pixel-accurate polygon outputs without pre-marking. Key highlights from Table 4:

- accuracy: our approach consistently delivers the highest vehicle instance mAP across both thresholds.
- efficiency: the system operates with multiply lower inference latency, thanks to architectural tuning and frame skipping.
- automation: the proposed model does not require any form of slot pre-labeling or manual intervention.

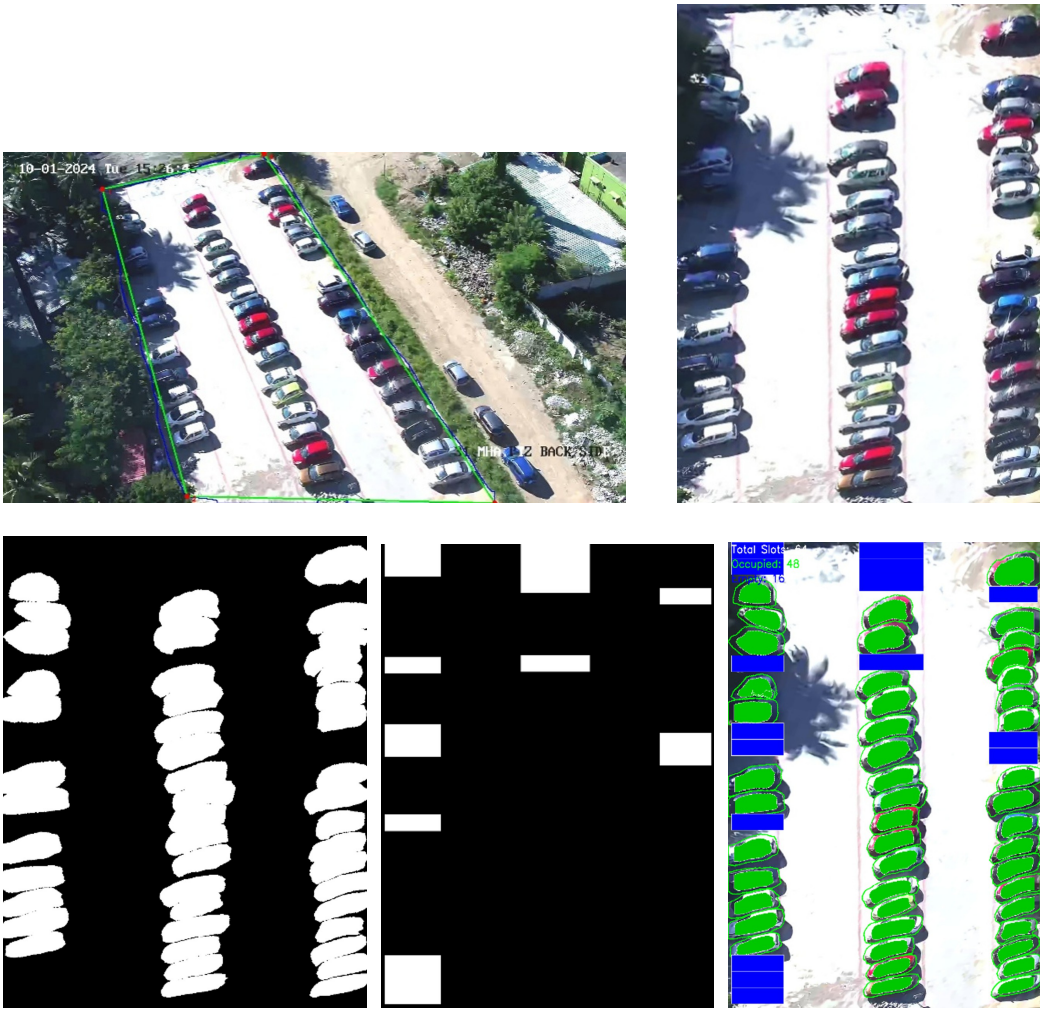


Figure 9: Pipeline: raw input, perspective transformation, intermediate masks/outputs, and final detection results (top row: inputs, bottom row: output variants).

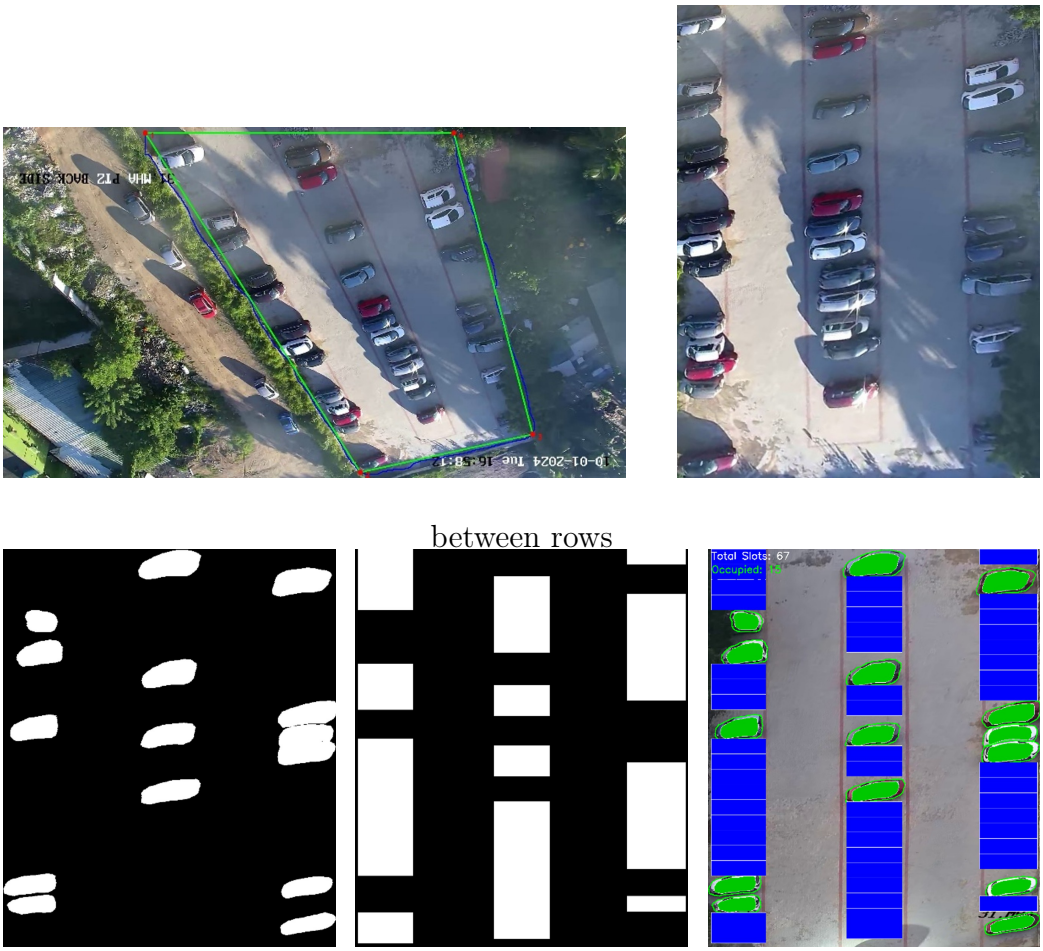


Figure 10: Flow of slot detection in varied light conditions.

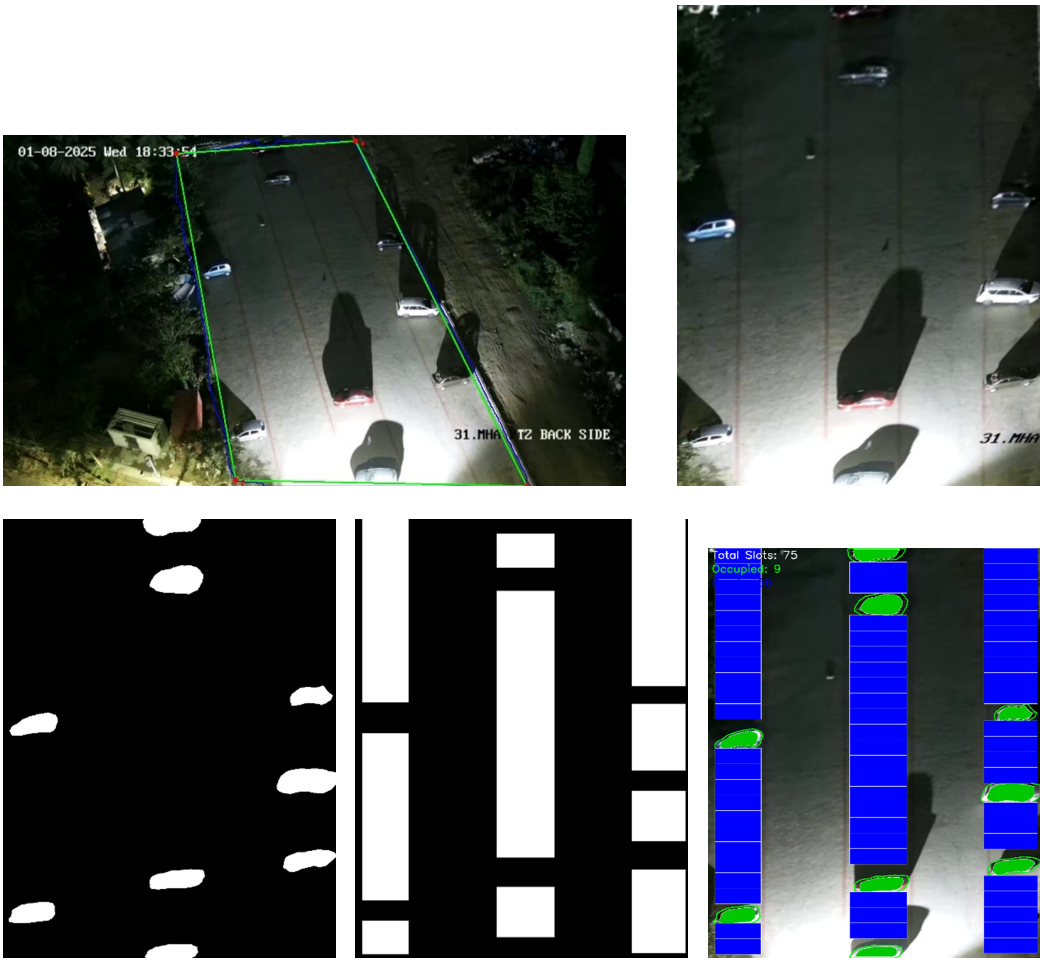


Figure 11: Flow of slot detection in low light conditions.

5.6. Ablation Study Analysis

Table 6 summarizes a series of ablation experiments. Removing either the GLCM texture fusion or dynamic slot finder sharply degrades slot and instance detection performance. Switching from polygonal decoding to simple bounding-box heads yields up to 7% mAP loss for strict IoU, confirming the decisive advantage of a polygon-based, hybrid-visual architecture.

- full system: swin+GLCM+Mask2Former+dynamic slot yields best mAP in Table 6.
- no GLCM: removing texture features reduces both thresholds by $\sim$ 3%–5%.
- bounding box only: polygon-free detection sharply reduces both pixel-wise and slot detection accuracy.
- no dynamic slot finder: using static pre-marked slots harms detection under heavy traffic/sparse conditions.

In summary, the proposed pipeline, see Figures 8, 10, 9, 7 and Tables 2, 3, systematically outperforms specialized and generic object detection models, achieving state-of-the-art parking slot detection for unmarked real-world lots.

6. Discussion: The Paradigm Shift to Map-Less Parking

The comparative analysis presented in Section 5.5 highlights a critical methodological divergence between the proposed HSM2F pipeline and the established benchmarks in parking detection, such as DeepPS (18), DCNN-Entrance Line (19), and approaches based on mAlexNet (5). While these competitors achieve high precision and recall metrics on predefined, static datasets, their core functionality is inherently restricted by the assumption of pre-marked, known slot geometry. Our work directly addresses the resulting limitations in deployability and flexibility by introducing a layout-free, map-less operational paradigm.

6.1. Redefining "Accuracy" in Real-World Deployment

The reliance of existing models on pre-marking forces integrators to manually survey, calibrate, and annotate the coordinates of every single parking slot before deployment. This introduces major bottlenecks in cost, time,

and system rigidity: if the parking lot geometry changes, the entire annotation process must be repeated. Our dynamic slot finder module, driven by the geometric analysis of detected car polygons in BEV, eliminates this dependency. Therefore, the system's primary benchmark shifts from simple numeric accuracy on a static test set to geometric and logical accuracy in a dynamic, unannotated domain.

- geometric superiority: we utilize the mAP_{75} metric (95.14%) which is a strict measure of pixel-level IoU and difficult for conventional bounding-box models to achieve. This focus validates the quality of our polygonal output, which is essential for metrically accurate gap analysis, a function bounding-box detectors fundamentally cannot perform.

- map-less metric: the ability to operate without pre-marking represents a qualitative leap that is difficult to capture in a standard quantitative table. We propose that successful dynamic slot inference, the capacity to correctly identify rows, gaps, and slot boundaries in a never-before-seen layout, is the decisive metric for next-generation systems.

6.2. The Necessity of the Hybrid Architecture

Our ablation study confirmed the joint necessity of the SwinMask2Former architecture elements:

- GLCM-FiLM fusion, contributing $\sim 4\%$ mAP_{75} gain: the performance degradation observed upon removing the GLCM features, where mAP_{75} drops from 95.14% to 91.20%, quantifies the value of texture robustness. Traditional CNNs often confuse shadows and wet pavement with vehicles. By injecting statistical texture data via the FiLM layer, our model enhances material selection, ensuring the system maintains high confidence even under severe glare and shadows: a persistent, real-world failure point for competing vision-only solutions.

- transformer global context: unlike models using localized receptive fields, the swin transformer's hierarchical global reasoning allows the system to comprehend the entire parking row and inter-car relationships. This contextual awareness is key to the success of the dynamic slot finder, as it allows the identification of geometric patterns such as parallel rows and clustering which are crucial for algorithmic tiling.

In conclusion, while the comparative tables establish the high performance and low latency of HSM2F relative to existing methods, the system's most valuable contribution lies in decoupling slot detection from layout rigidity. This shift, enabled by pixel-accurate instance segmentation and dynamic geometric reasoning in BEV, makes the proposed pipeline scalable, flexible, and truly adaptable to the complex, uncontrolled environments of smart city infrastructure.

7. Future Scope

While the proposed system establishes a high-performance baseline for vision-based parking analysis, several promising avenues remain for further strengthening and extending its capabilities:

- performance optimization: applying model quantization such as TensorRT, distillation, and dynamic frame adaptation may further minimize latency and computational load, making the solution viable for resource-constrained edge devices or embedded systems.
- cross-dataset generalization: extensive benchmarking on challenging, unseen datasets such as PKLot or CNRPark will help assess and improve generalizability across domains, lighting, and surveillance setups.
- research extensions: future efforts may include multi-modal fusion by combining vision with low-cost sensors, self-supervised adaptation to new lots, temporal slot prediction, and integration with intelligent traffic and navigation systems.

By systematically addressing these areas, the framework can evolve into the cornerstone of next-generation smart city mobility platforms and urban parking intelligence.

8. Conclusion

This work presented a robust layout-free parking vacancy detection framework, leveraging advanced vision transformers, GLCM-based texture fusion, and an instance segmentation driven end-to-end pipeline. By combining a swin transformer backbone with a SwinMask2Former instance head and data-driven GLCM texture features, the system achieved high-precision, polygonal detection of vehicles in diverse environments. The introduction of the

`DynamicSlotFinder` module enables map-less, fully automated slot vacancy inference by algorithmically analyzing spatial gaps and vehicle groupings in a perspective-corrected BEV. Comprehensive quantitative and qualitative evaluations demonstrate the performance across multiple benchmarks. The modularity of the approach, its resistance to visual noise, and its real-time inference capability with frame-skip and slot tracking collectively address the challenges of static, sensor-reliant, or hard-wired vision systems. The pipeline's scalability and flexibility suggest a strong applicability for real-world deployment, urban planning, and future smart parking infrastructure.

References

[1] J. A. Vera-Gómez, A. Quesada-Arencibia, C. R. García, R. S. Moreno, and F. G. Hernández, "An intelligent parking management system for urban areas," *Sensors*, vol. 16, no. 6, p. 931, 2016.

[2] A. Dahiya, P. Mittal, Y. K. Sharma, U. K. Lilhore, S. Simaiya, E. Ghith, and M. Tlija, "Machine Learning-Based Prediction of Parking Space Availability in IoT-Enabled Smart Parking Management Systems," *Journal of Advanced Transportation*, vol. 2024, 2024.

[3] J. Arjona, M. Linares, J. Casanovas-Garcia, and J. J. Vázquez, "Improving parking availability information using deep learning techniques," *Transportation Research Procedia*, vol. 47, pp. 385–392, 2020.

[4] M. Heimberger, J. Horgan, C. Hughes, J. McDonald, and S. Yogamani, "Computer vision in automated parking systems: Design, implementation and challenges," *Image and Vision Computing*, vol. 68, pp. 88–101, 2017.

[5] G. Amato, F. Carrara, F. Falchi, C. Gennaro, and C. Vairo, "Car parking occupancy detection using smart camera networks and deep learning," in *2016 IEEE symposium on computers and communication (ISCC)*, 2016, pp. 1212–1217.

[6] A. Farley and H. Ham, "Real time IP camera parking occupancy detection using deep learning," in *Procedia Computer Science*, vol. 179, pp. 606–614, 2021.

[7] X. Ding and R. Yang, "Vehicle and Parking Space Detection Based on Improved YOLO Network Model," in *Journal of Physics: Conference Series*, vol. 1325, no. 1, p. 012185, 2019.

[8] J. J. J. De Ala and K. C. Faelangca, "Real-Time Monitoring Of Parking Lot Space Detection," *Educational Administration: Theory and Practice*, vol. 30, no. 5, pp. 11268–11285, 2024.

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2017, pp. 2961–2969.

[10] D. Bolya, C. Zhou, F. Follmer, and J. Lee, "YOLACT: Real-time instance segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*, 2019, pp. 9157–9166.

[11] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*, 2021, pp. 10012–10022.

[12] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-attention mask transformer for universal image segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2022, pp. 1290–1299.

[13] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on systems, man, and cybernetics*, no. 6, pp. 610–621, 1973.

[14] A. P. Irfan, N. Nur, and F. Wajidi, "Parking Slot Detection Using GLCM and Similarity Measure," in *IOP Conference Series: Materials Science and Engineering*, vol. 875, no. 1, p. 012028, 2020.

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *arXiv preprint arXiv:2010.11929*, 2020.

[16] L. Encío, C. Díaz, C.R. del-Blanco, F. Jaureguizar, and N. García, "Visual Parking Occupancy Detection Using Extended Contextual Image Information via a Multi-Branch Output ConvNeXt Network," *Sensors*, vol. 23, no. 1, p. 383, 2023.

[17] Bhattiprolu, S. (n.d.). bnsreenu/digitalsreeni-image-annotator: A python based GUI to annotate images and save annotations as COCO style JSON

format [Software]. GitHub. https://github.com/bnsreenu/digitalsreeni-image-annotator

[18] Z. Zhang, L. Shen, Y. Zhang, H. Zhao, and S. Yang, "DeepPS: A deep learning-based parking slot detection method using convolutional neural networks," *Journal of Visual Communication and Image Representation*, vol. 54, pp. 148–158, 2018.

[19] W. Li, H. Cao, J. Liao, J. Xia, L. Cao, and A. Knoll, "Parking Slot Detection on Around-View Images Using DCNN," *Frontiers in Neurorobotics*, vol. 14, p. 46, 2020.