Journal of Artificial Intelligence and Data Science (JAIDA)
e-ISSN: 2791-8335
https://dergipark.org.tr/pub/jaida

Vol. 5, No. 1, pp. 12-27, 2025
Research Article
Received: 05/01/2025 Accepted: 26/06/2025

Real-Time Table Occupancy Detection in Restaurants Using Object Detection and Computer Vision Techniques

Ali Kerem GÜLER [1,*], Ali MUSA [2]

Abstract
This work presents an end-to-end system to monitor, analyze, and forecast table occupancy in a restaurant setting using computer vision and time-series analysis. The methodology first involves training a YOLOv8 object detection model on a custom-annotated image dataset to achieve high-precision table detection. The system then determines the number of people at each table and logs this information to construct a time-series dataset. Subsequently, this dataset is used to train and compare predictive models—specifically Exponential Smoothing and Random Forest—to forecast future occupancy patterns. The custom-trained detection model demonstrated outstanding performance, achieving a Mean Average Precision (mAP@.5:.95) of 0.985 for table detection. For the forecasting module, the Exponential Smoothing model proved more robust, predicting occupancy with a Mean Absolute Error (MAE) of 1.23, outperforming the Random Forest model. By integrating high-accuracy real-time detection with predictive analytics, this study provides restaurant operators with a powerful tool for data-driven operational planning, such as optimizing staff allocation and inventory management, ultimately aiming to enhance service quality and customer satisfaction.
Keywords: Object Detection, Image Processing, Computer Vision, Time-Series Forecasting, Yolov8, Exponential Smoothing, Random Forest.

1. Introduction

The restaurant industry continuously seeks innovative solutions to maximize customer satisfaction and enhance operational efficiency [1, 2]. The ability to accurately monitor the number of customers seated at each table and determine the overall occupancy rate can significantly improve a restaurant's operations. This information not only helps in optimizing wait times and table turnover rates to enhance customer service but also allows for more efficient resource allocation, such as staff and inventory management.

Traditional methods for monitoring table occupancy typically rely on manual counts by staff or simple reservation systems. However, these approaches are time-consuming and prone to errors. Moreover, manually monitoring table occupancy is not always a reliable method. Through computer vision and image processing approaches, machines can analyze visual data from their surroundings to detect and categorize objects [3 – 8]. These approaches enable the detection of objects such as tables and chairs in restaurants, and the analysis of these objects can be used to assess table occupancy.

This study introduces an intelligent system that employs the YOLO algorithm, an advanced object detection tool, to evaluate table occupancy in restaurants [9 – 12]. The system's pipeline begins with the detection of tables using a model fine-tuned on a custom dataset, which is specifically designed to overcome the challenges of varied lighting and table designs in a restaurant environment. Subsequently, it implements a dynamic Area of Interest (AOI) mechanism to isolate the relevant table area and accurately count the individuals present. All occupancy metrics are then logged with timestamps to construct a time-series dataset for further analysis. This study makes several key technical contributions to address the challenges of restaurant occupancy monitoring:

- Studies in the literature generally explore occupancy detection in various contexts such as parking lots, rooms, and public spaces using computer vision techniques and deep learning models. However, they do not specifically address the issue of table occupancy detection in restaurant environments. In this study, image processing techniques have been applied specifically to detect occupancy in restaurants.
- Instead of using ready-made datasets, special table and chair datasets have been created from different angles and under different lighting conditions in a pilot restaurant environment designated for the study.
- A specialized object detection model was developed and validated by fine-tuning the YOLOv8 architecture on a novel, custom-annotated dataset collected from a real-world restaurant environment.

*Corresponding author

Ali Kerem GÜLER; Protel A.Ş., İstanbul, Türkiye; e-mail: aguler@protel.com.tr; 0000-0003-3405-005X

Ali MUSA; Protel A.Ş., İstanbul, Türkiye; e-mail: amusa@protel.com.tr; 0009-0000-5829-5698

- A dynamic restaurant table detection and isolation mechanism was implemented, utilizing a geometric algorithm based on Euclidean distance. This method ensures that each camera reliably focuses on its designated primary table, even in views containing multiple objects, thereby preventing data ambiguity and enhancing the accuracy of the subsequent people detection module.
- The system converts raw occupancy data into a structured time-series dataset and applies predictive models, such as Exponential Smoothing and Random Forest, to generate actionable forecasts. This transforms the system from a mere monitoring tool into a data-driven decision support system for operational planning.
- A deployable system architecture is presented, featuring automated data logging via a cron-scheduled API and a structured error-tracking mechanism. This demonstrates the practical applicability of the proposed solution for real-world restaurant operations.

Advancing technology can help restaurants reduce the time and labor spent on manually monitoring table occupancy while also enhancing operational efficiency. The implementation of the YOLO algorithm in this context offers several advantages. YOLO's ability to process images quickly and accurately makes it ideal for real-time applications in a restaurant's dynamic environment. Additionally, this system provides restaurant operators with comprehensive insights into their operations, which can contribute to data-driven decision-making and increased customer satisfaction. However, computer vision algorithms do have some limitations. For instance, in poorly lit areas or when other objects obstruct the view, detecting objects can be challenging. This underscores the need for regular checks of the system's detection accuracy.

In conclusion, computer vision algorithms offer a valuable solution for restaurants seeking to reliably and effectively verify table occupancy rates. While the technology is not without its imperfections, it provides substantial support in improving the operational efficiency and customer satisfaction of restaurants.

The remainder of this paper is structured as follows: The Literature Review section presents a review of the literature on table occupancy monitoring and the YOLO algorithm. The Methodology section details the design and implementation of the proposed YOLO-based intelligent table occupancy monitoring system. In the Results and Discussion section, the performance of the system is evaluated, and the results are discussed. Finally, the Conclusion section summarizes the findings of the study and outlines directions for future research.

2. Literature Review

In this study, it is important to understand the broader context of occupancy detection and related techniques. In the literature, there are various studies and reviews conducted for detecting and managing occupancy in various settings such as parking lots, rooms, and public spaces. Although these studies do not specifically focus on restaurant table occupancy, they provide some useful insights that can be applied in this research.

Research into parking guidance systems and parking space sensing demonstrates the potential of using image processing techniques and deep learning models to manage occupancy in complex environments with varying weather and lighting conditions [14 – 19]. Techniques such as manual seeding, boundary search, image difference, object detection, edge detection, voting algorithms, image converters and convolutional neural networks have been implemented to improve detection accuracy and real-time performance [13, 15 – 18].

Studies on room occupancy detection and people counting in indoor environments further demonstrate the efficiency of image-based methods in tracking multiple people with different heights and movements. Unsupervised real-time methods and edge-based transfer learning frameworks have been proposed to provide accurate occupancy detection in various indoor environments without requiring special training steps [20, 21].

Seat occupancy sensors, including capacitive and IR sensors, have been developed to accurately distinguish human occupancy from inanimate objects [22]. These low-cost, easy-to-install sensors provide valuable information that can be applicable to restaurant table occupancy detection for occupancy detection in public spaces with fixed seating arrangements.

Studies on YOLO-based object detection are prevalent in the literature. However, research focusing on table occupancy, people counting, and occupancy monitoring in restaurant settings is comparatively limited. YOLO-based solutions are utilized to track human movement and spatial occupancy rates in real-time across various indoor and outdoor environments, including restaurants. Particularly in the post-pandemic era, vision-based solutions have become crucial for restaurants to effectively manage capacity and maintain social distancing. The YOLO algorithm is often favored in these applications for its real-time performance and high accuracy [23].

Restaurants can significantly enhance their operational efficiency by monitoring customer volume and peak hours. Image processing-based people counting systems aid in personnel and resource planning by identifying a restaurant's busiest periods. Through the analysis of camera footage, these applications provide data on when the establishment is most crowded, thereby improving table management and increasing customer satisfaction [24].

In summary, the literature review confirms that computer vision, particularly deep learning models like YOLO, is highly effective for various occupancy detection tasks. However, the reviewed studies also reveal two key insights that directly support the methodology employed in this study. First, the success of these models is often contingent upon domain-specific datasets, as general-purpose models struggle with unique environmental challenges such as varying lighting conditions and object types. Second, while sensor-based approaches demonstrate high accuracy for fixed seating, they lack the flexibility required for dynamic restaurant layouts and cannot provide the rich visual context offered by a camera-based system. This study advances table occupancy detection in restaurant environments by proposing a module for demand forecasting and operational decision support. Through the integration of time-series analysis with real-time data, the study aims to enable restaurants to conduct more effective inventory and resource management. In this respect, the study contributes to the literature on restaurant-focused operational planning.

3. Methodology

Based on the gaps and opportunities identified in the literature, the methodology for this study was designed around a set of selected techniques to ensure both high performance and practical relevance. For the object detection task, the YOLOv8 architecture was chosen as the core component. As a state-of-the-art, single-stage detector, it provides a high-level balance between processing speed and detection accuracy, a critical requirement for potential real-time restaurant applications. Its anchor-free design offers high performance on objects of varying sizes and shapes, which is directly applicable to the diverse table types found in restaurants. For the forecasting module, a comparative approach was adopted. Exponential Smoothing (Holt-Winters) was selected as a powerful statistical baseline, given its proven effectiveness in modeling time-series data with strong trend and seasonality components, which are characteristic of customer traffic in a restaurant. To challenge this classical model, a Random Forest Regressor was implemented. As a robust machine learning ensemble method, it was chosen to investigate whether complex, non-linear relationships between engineered temporal features (e.g., hour of day, day of week) could yield more accurate occupancy predictions.

In this section, a methodological approach is presented in which the occupancy information of tables monitored through cameras is calculated using object detection algorithms. Additionally, this approach aims to create time-series data by recording metrics such as table occupancy rate, number of people, and table ID based on requests received at specific intervals.

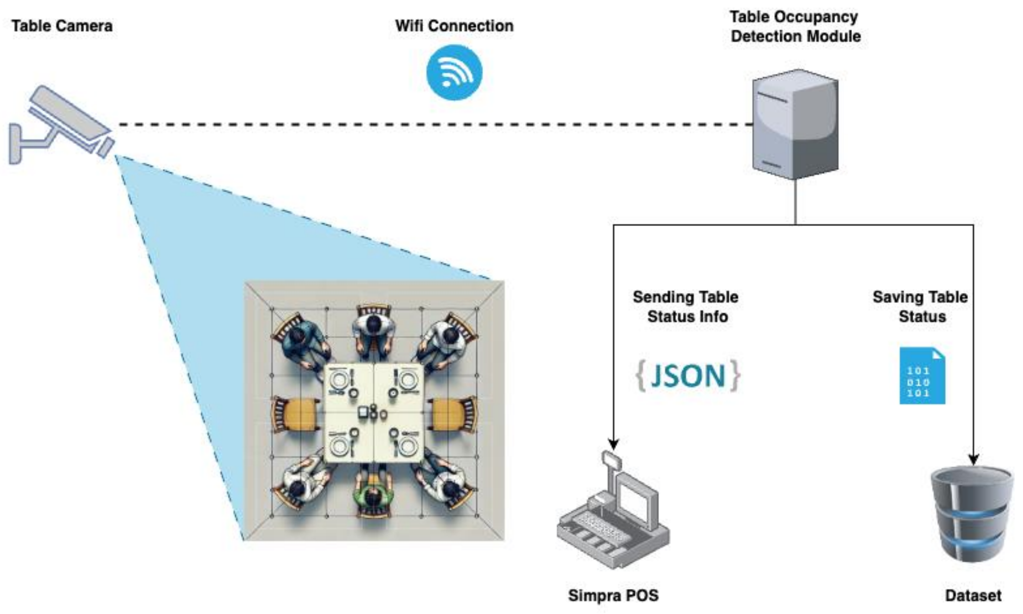

Figure 1. *Table Occupancy Detection System Architecture.*

The tables are monitored from above through cameras (as shown in Figure 1). The camera images are received by the table occupancy detection module, which works on a remote server via a Wi-Fi connection. As a result of the analyses conducted on the images, the table occupancy information is sent to both the POS application and the database for dataset creation.

3.1. Dynamic Table Detection

Each environment has varying conditions, such as different color saturation levels and light intensity. Moreover, objects such as tables have various colors, shapes and sizes. Pre-trained object detection algorithms are trained with a large amount of images containing numerous class labels. This allows for the rapid use of these models by simply providing an image or video. However, due to changing environmental conditions and the diverse colors and features of objects, pre-trained models may not always achieve the expected performance in recognizing the desired classes. This necessitates training the model with custom data. Therefore, a custom image dataset (shown in Figure 2) consisting of tables with varying numbers and colors was created for the restaurant environment set up for the study. The dataset includes image data of tables of different sizes and shapes from both the indoor and outdoor areas of the restaurant.

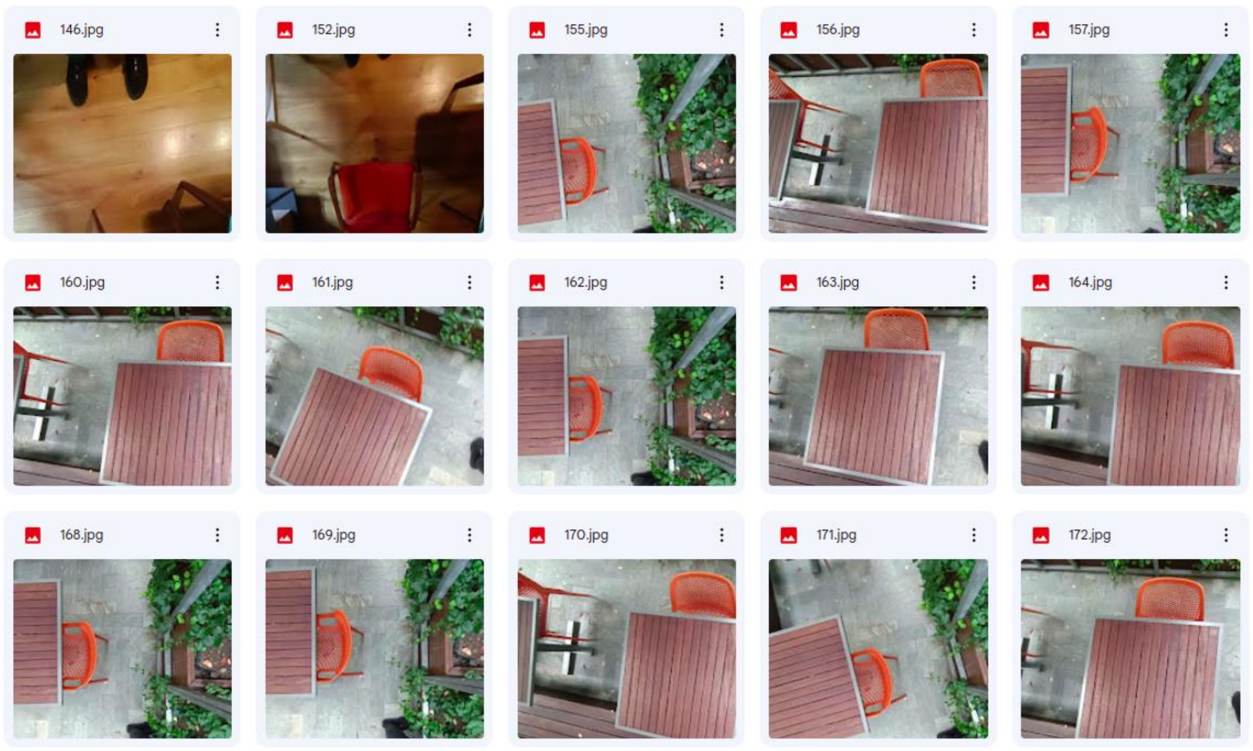

Figure 2. *Custom table dataset created from test restaurant environment.*

The custom table dataset contains a total of 792 images. The dataset was gathered by extracting frames from videos collected in the restaurant environment. These frames were obtained and saved using the OpenCV library, which is compatible with the Python programming language [25]. OpenCV (Open-Source Computer Vision Library) [26] is an open-source library that provides various functions for applications such as image and video analysis, object detection, face recognition, and motion tracking. To ensure high accuracy in the object detection algorithm, frames were extracted, and those with appropriate quality and clarity were selected. To achieve high accuracy in the object detection algorithm, suitable frames were selected from the extracted ones. For the object detection algorithm to learn from the data, it requires the objects in the images along with their corresponding labels. To this end, a labeling process was performed on the image data. The labeling process was carried out using an open-source tool called LabelImg and a total of 792 images were labeled [27].

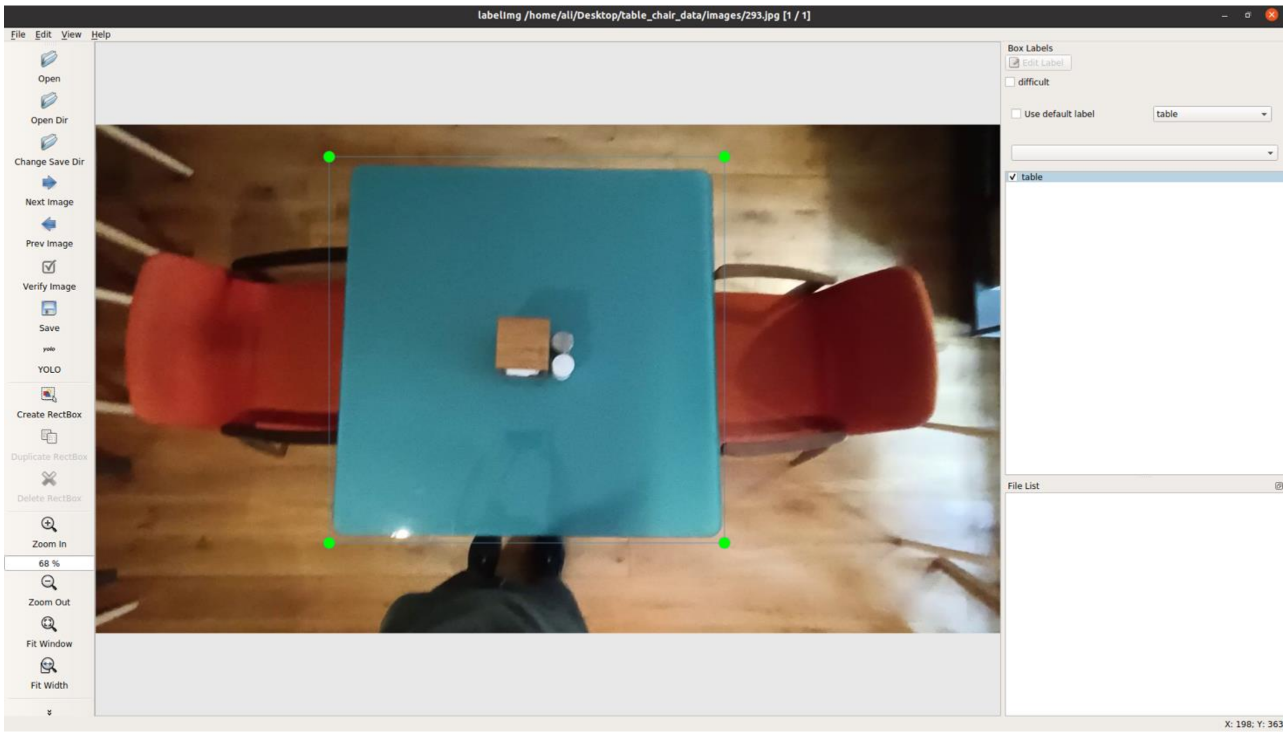

Figure 3. *An image illustrating the labeling process performed on one of the samples in the dataset using the LabelImg tool*.

Using the LabelImg tool, the tables in the images were labeled (shown in Figure 3) with the table class label. For each image, the coordinate and label information were saved in .txt format. After the image data and their corresponding labeled data were generated, the preprocessing steps required for training the object detection algorithm were completed.

3.2. Object Detection Model Training

YOLOv8, the eighth version of YOLO, was utilized on a custom table dataset specifically created for model training. YOLOv8 is an anchor-free, single-stage object detection model designed to detect and classify objects in an image in real time [12]. Unlike previous versions, YOLOv8 uses a more streamlined architecture with dynamic modules that allow flexibility across diverse datasets and use cases.

YOLOv8 uses an object detection algorithm that effectively identifies objects in an image by dividing the object detection process into fundamental steps. This process includes operations such as feature extraction, grid-based prediction, bounding box regression, and classification. The object detection algorithm first extracts features from the input image. These features are represented by tensors that specify the height, width, and the number of channels of the image. Then, a deep convolutional neural network (CNN) processes the image to generate feature maps. YOLOv8 divides the image into a grid. Each grid cell predicts bounding boxes and class probabilities [28]. For each cell, the model outputs predictions (Eq. 1), each containing bounding box coordinates b_i, an objectness score o_i, and class probabilities c_i.

$$P = \{(b_i, o_i, c_i) | i = 1, \dots, n\} \tag{1}$$

The bounding box parameters are predicted relative to the grid cell. The exact bounding box coordinates are computed in Eq. 2. The center of the bounding box is determined by the x-coordinate $(\sigma(t_x) + c_x)$ and the y-coordinate $(\sigma(t_y) + c_y)$. These coordinates are calculated by adding the predicted offset values (t_x, t_y), compressed using the sigmoid function, to the top-left corner coordinates of the grid cell (c_x, c_y). The width $(e^{t_w} \cdot w_a)$ and height $(e^{t_h} \cdot h_a)$ of the bounding box are obtained using the scaling factors (t_w, t_h), which are exponentially predicted based on the width and height of the anchor box (w_a, h_a). This regression process allows the model to refine the location and size of the detected objects, enabling precise object detection.

$$\widehat{b_i} = \left(\sigma(t_x) + c_x,\ \sigma(t_y) + c_y, e^{t_w} \cdot w_a + e^{t_h} \cdot h_a\right) \tag{2}$$

YOLOv8 eliminates the reliance on predefined anchor boxes, reducing computational overhead and improving detection accuracy, especially for small or overlapping objects. The model dynamically adjusts feature aggregation for optimal performance, making it adaptable to various detection tasks. Additionally, YOLOv8 applies Neural Architecture Search (NAS) to optimize its architecture automatically, ensuring a better trade-off between speed and accuracy [29].

After performing the model training process, the weights of the model that showed the best performance on the validation set at each epoch result were saved. Afterwards, the detection performance of the best model was observed with test images.

3.3. Midpoint and Distance Calculation Object

Cameras placed in the restaurant environment can also capture other tables within their field of view. Consequently, the object detection algorithm can detect multiple tables in a single frame. However, the cameras, positioned to view the tables from a top-down perspective at a right angle, are not mounted at a sufficiently high altitude. This can lead to overlapping and confusion between table areas when detecting occupancy within a two-dimensional space containing multiple tables. As shown in Figure 4, the table areas detected within the bounding boxes are very close to one another and almost overlap. Since this can cause challenges during the analysis phase, a per camera per table approach was adopted. In this approach, each camera is dedicated to monitoring a single table.

To detect the table closest to the camera, the Euclidean distance approach was applied to calculate the distances of objects from the center point of the image frame. For this process, the coordinates of the center pixel of the image frame and the center pixel coordinates of the detected table objects were first determined. Subsequently, the distances between the x and y coordinates of the center points of the detected table objects and the center pixel of the image frame were calculated using the Euclidean distance method. As a result of this calculation (Algorithm 1), the coordinates of the point closest to the center pixel were identified. This point also represents the table closest to the center point, where occupancy detection will be performed.

Algorithm 1: *Finding the Closest Point to a Central Point*

```
Input: A list of 2D points coordinate $points\_arr = [(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)]$
$points\_arr = [(x1, y1), (x2, y2), \dots, (xn, yn)]$
Output: The index of the point closest to the central point $cp = (320, 320)$
   1. Define $cp \leftarrow (320, 320)$.
   2. Initialize an empty list $dist\_arr \leftarrow []$.
   3. For each $point = (x, y)$ in $points\_arr$:
         a. Compute the Euclidean distance:
               $dist \leftarrow \sqrt{(x - cp_x)^2 + (y - cp_y)^2}$
         b. Append $dist$ to $dist\_arr$.
   4. Find the index of the minimum value in $dist\_arr$:
               $index \leftarrow argmin(dist\_arr)$

   5. Return $index$.
```

3.4. Setting the Area of Interest

After identifying the table closest to the center point, the area of interest was adjusted to ensure the efficient operation of the human detection module. With this adjustment, instead of performing human detection across the entire image, occupancy detection was conducted solely within the area containing the relevant table. However, as observed in Figure 4, some chairs were located outside the bounding box of the table designated for occupancy detection. Since this bounding box also serves as the area of interest, adjustments were necessary to ensure that individuals using the table were fully visible in the analyzed image. To include the chairs within the area of interest, a padding operation was applied, expanding this area by 30 pixels horizontally and vertically.

3.5. Human Detection Module

After the padding operation, a pretrained YOLOv8 model was used to detect individuals utilizing the table within the area of interest. This model had been pretrained on the COCO (Common Objects in Context) dataset, which includes images from various categories. In this study, the pretrained model was configured to detect only the classes belonging to the human category [30].

3.6. Calculating Occupancy Metrics

The occupancy information is calculated based on the number of chairs at the table and the number of individuals detected within the padded area of interest. The table occupancy data includes information such as the timestamp, the number of individuals, and the occupancy rate. Since these data points contain timestamps, they can be transformed into a time-series dataset, enabling tracking and time-based analyses in line with the operational plans of the enterprise.

Table 1. *A table containing real-time table occupancy information for two different tables.*

timestamp	table id	table cam status	chairs count	people count	people ratio
2024-05-22 08:00	table_01	active	4	0	0.0
2024-05-22 08:00	table_02	active	4	1	0.25

Table 1 presents the real-time occupancy information for two separate tables monitored by two different cameras. In addition to the table occupancy data, information related to each table and the corresponding monitoring camera is also saved.

3.7. Logging and Time Series Dataset

In this study, Cron, a time-based job scheduler available in Unix-like operating systems, was utilized. A crontab, which is a table containing Cron jobs (scheduled tasks for automation), was initiated to run the application at specified intervals (e.g., every five minutes) [31]. This process enabled the system to continuously update the specified log file, facilitating real-time monitoring and data collection.

An API was developed for the application using the Flask web framework. Flask is a Python-based web framework commonly used to create APIs (Application Programming Interface) that facilitate data stream [32]. When a request is received via the API, a health check is first performed on the connected camera systems. If any camera is detected to be offline, the occupancy information for the table within that camera's field of view is set to zero and saved in the log file accordingly. If the cameras are active, the table occupancy data in the log file is updated with the data received from the table occupancy module. Additionally, through the API, it is possible to retrieve either the latest log entry or all occupancy information from the log file. The API directs incoming requests to the appropriate endpoint and based on the request structure, returns either all log entries or only the most recent log registry.

Finally, an error tracking system was implemented, where error logs are stored in a log file. All errors occurring while the API is in use are automatically saved in this file. This enables the effective identification of the root causes of issues, making their resolution possible.

3.8. Occupancy Forecasting Module

Leveraging occupancy data collected through camera-based computer vision techniques were constructed a detailed time-series dataset encompassing 1,344 records, collected at 15-minute intervals for two distinct tables. Each record includes a timestamp, table ID, number of occupants (people count), and occupancy ratio.

Table 2. *Distribution of the People Count in the Dataset.*

People Count	Number of Observations	Percentage (%)
0	286	21.28
1	266	19.79
2	261	19.42
3	266	19.79
4	265	19.72

Initially, comprehensive exploratory data analysis (Table 2) and preprocessing steps, including handling missing values, correcting timestamps, resampling the data, and seasonality analysis were performed. Subsequently, daily and hourly occupancy heatmaps were generated to visualize occupancy patterns clearly and identify peak usage periods effectively.

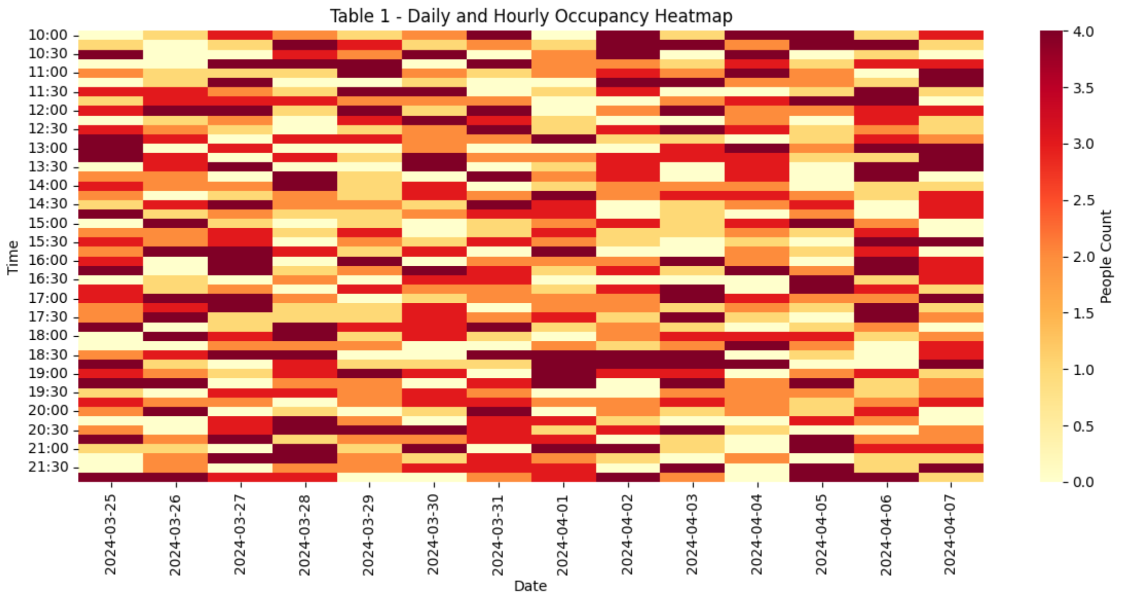

Figure 4. *Heatmap of daily and hourly occupancy for first table of the restaurant (visualized in red tones).*

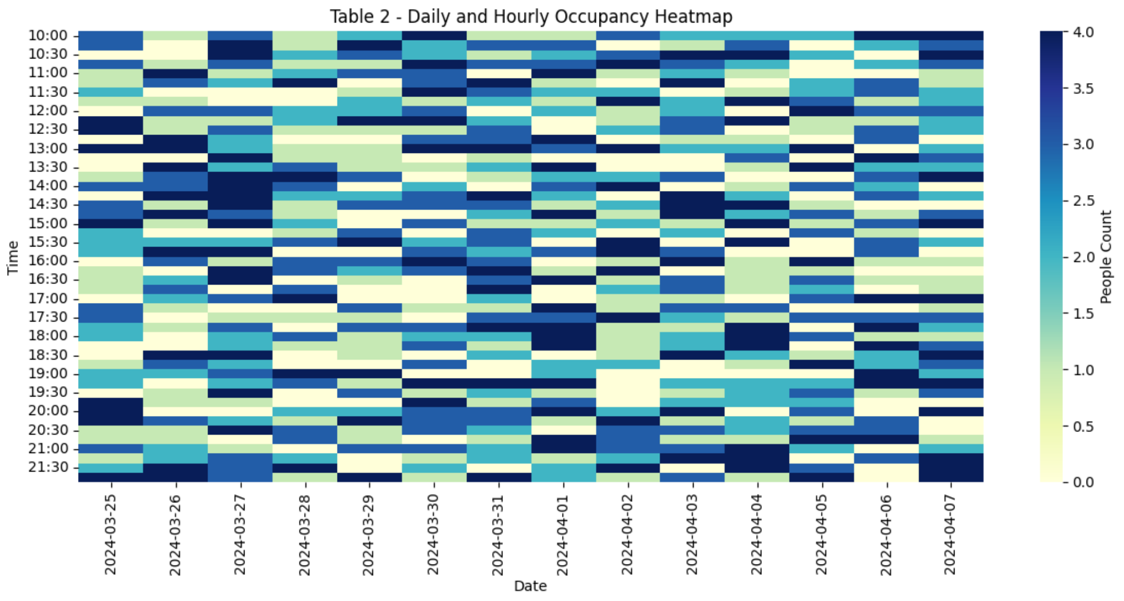

Figure 5. *Heatmap of daily and hourly occupancy for second table of the restaurant (visualized in blue tones).*

A heatmap analysis was conducted to visualize the daily and hourly occupancy data for two different tables within the restaurant. This analysis aims to reveal on which days and during which time intervals the tables are most frequently occupied. For first table of the restaurant (Figure 4), the heatmap analysis indicates that occupancy increases notably during lunch hours (12:00 - 14:00) and evening hours (18:00 - 20:00). Furthermore, the occupancy rate was found to be higher during evening hours on weekends (Saturday-Sunday). For second table of the restaurant (Figure 5), the analysis shows a similar increase in occupancy during lunch and evening hours. The overall occupancy level for second table of the restaurant was observed to be more balanced on weekdays and higher on weekends. These heatmap visualizations can serve as a decision support tool, enabling restaurant management to organize operations based on customer demand. In addition, operational decisions such as staff scheduling, stock management, and inventory control can be made more efficiently in light of these analyses.

Subsequently, the dataset was splitted into a training set (90%) and a test set (10%). Two predictive models were trained on the training dataset to forecast table occupancy. These models were Exponential Smoothing and

Random Forest [33, 34]. The Exponential Smoothing model was configured with additive trend and seasonality components, along with seasonal periods adjusted to capture daily patterns (seasonal_periods=48). For the Random Forest Regressor, additional feature engineering was applied, leveraging temporal features such as the hour of the day and the day of the week. The model was configured with 100 estimators and a fixed random state to ensure reproducibility.

4. Results and Discussion

In this section, the results of the study on the implementation of the smart table occupancy monitoring system in the restaurant environment and occupancy forecasting module are presented and the findings are discussed. The performance of the YOLOv8 model, fine-tuned on a custom image dataset collected from a restaurant, was evaluated on the validation set using standard object detection metrics. The results in Table 3 indicate that the custom-trained model achieved a Mean Average Precision (mAP) of 0.997 for the "table" class at an IoU threshold of 0.5 (mAP@.5). This high score demonstrates the model's strong ability to accurately identify and localize restaurant tables within the images. Furthermore, the model achieved a Recall of 1.0, indicating that it successfully detected every table instance in the validation set with low false negatives. A Precision score of 0.998 signifies that the model's detections are almost entirely correct, with a negligible false positive rate. For table detection, a high mAP@.5:.95 score of 0.985 was achieved. This metric evaluates performance across a stricter range of IoU thresholds (0.5 to 0.95) and signifies that the model's bounding box predictions are not only correct but also highly precise in their placement. This high degree of localization accuracy confirms the successful definition of the Area of Interest (AOI), which is crucial for the subsequent stage of people detection.

Table 3. *YOLOv8 Model Performance Metrics on the Custom Validation Set.*

Class	Precision (P)	Recall (R)	mAP@.5	mAP@.5:.95
table	0.998	1.0	0.997	0.985

Figure 6 demonstrates the detection results visualized on an image provided to the model as test data. These results demonstrate that the object detection model can recognize tables of various sizes, shapes, and colors under different lighting conditions. The performance of the table occupancy detection system depends on the accurate detection of individuals at each table. Since the area where occupancy is detected relies on the results of the object detection algorithm, the success of this algorithm directly impacts the overall system performance.

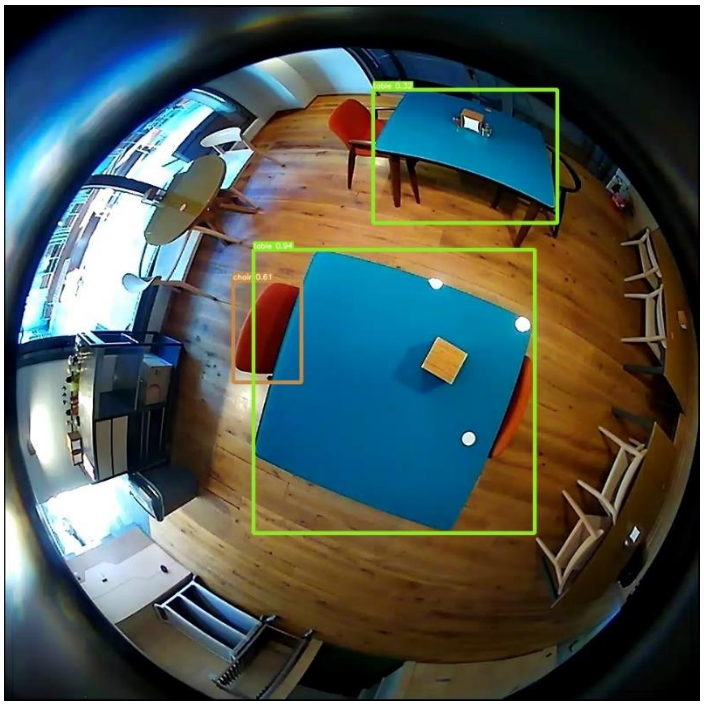

Figure 6. *Object detection model result on sample image data.*

The integration of center point and distance calculation functions has enabled each camera to focus solely on the table closest to it. This process prevented confusion caused by overlapping table areas. Additionally, to enhance the human detection module's ability to identify individuals within the table area, a padding method was applied, extending the detection area horizontally and vertically. This adjustment ensured that chairs were also included in the table area. As a result of these processes, the detection area (area of interest) on the test image is presented in Figure 7.

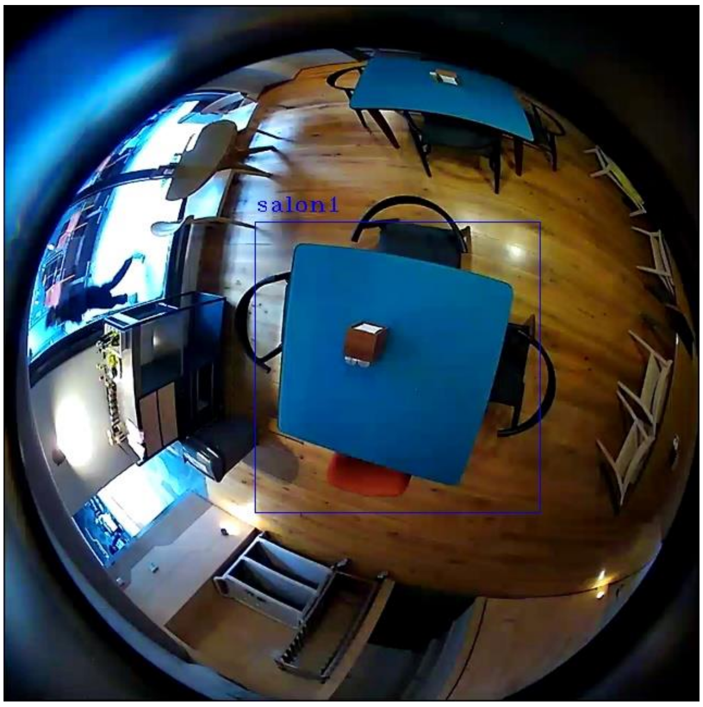

Figure 7. *Dynamic table detection.*

To evaluate the human detection module's ability to recognize individuals, some test subjects were placed within the designated area of interest and used the table. During this process, test images were captured and presented to the human detection module. It was observed that the module detected individuals within the area. The visualization of human detection results on a test image is shown in Figure 8. The human detection module established the relationship between the number of detected individuals and the table occupancy rate, providing this information to the table occupancy detection system.

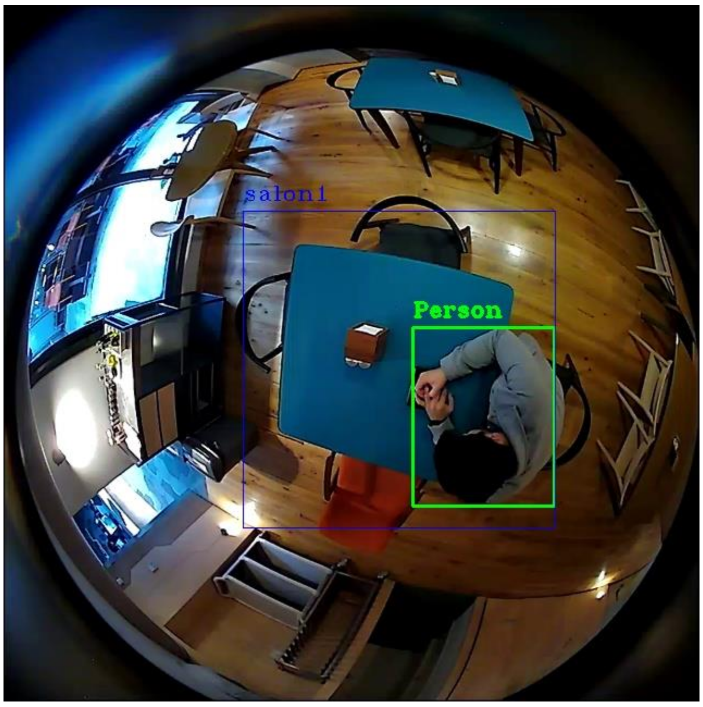

Figure 8. *Detection of people in the region of interest with the Human Detection module.*

Subsequently, the table occupancy detection system utilized the information received from the human detection module to save data such as timestamp, number of individuals, and occupancy rate into a log file at specified time intervals. Table usage information at specific intervals was saved in the log file through the scheduling of Cron to execute the application at periodic intervals. This periodic data collection and saving process provided a dynamic dataset, enabling the monitoring of occupancy rates and usage patterns over time.

To evaluate the table usage information saved in time-series format by the table occupancy detection system, log entries from about an hour and a half time interval were analyzed. Camera recordings from the same time interval was also manually reviewed. The comparison of both results is presented in Table 4. The analysis revealed that the table occupancy detection system accurately identified table usage within the specified time intervals. As a result, the methodology implemented in this study provides a comprehensive approach to monitoring and analyzing tables in a restaurant environment. By utilizing object detection algorithms, human detection modules, and logging mechanisms, time-series data on table occupancy were generated. This time-series data can contribute to businesses making more efficient decisions and gaining valuable insights for their operational processes.

Table 4. *Comparison of Table Occupancy system and Manual Analysis results.*

	Table Occupancy System Results				**Manuel Analysis Results**			
	table_01		**table_02**		**table_01**		**table_02**	
timestamp	people count	people ratio	people count	people ratio	people count	people ratio	people count	people ratio
2024-03-24 15:35	0	0.0	0	0.0	0	0.0	0	0.0
2024-03-24 15:40	0	0.0	0	0.0	0	0.0	0	0.0
2024-03-24 15:45	0	0.0	0	0.0	0	0.0	0	0.0
2024-03-24 15:50	1	0.25	0	0.0	1	0.25	0	0.0
2024-03-24 15:55	1	0.25	0	0.0	1	0.25	0	0.0
2024-03-24 16:00	0	0.0	0	0.0	0	0.0	0	0.0
2024-03-24 16:05	0	0.0	0	0.0	0	0.0	0	0.0
2024-03-24 16:10	0	0.0	1	0.25	0	0.0	1	0.25
2024-03-24 16:15	0	0.0	1	0.25	0	0.0	1	0.25
2024-03-24 16:20	0	0.0	0	0.0	0	0.0	0	0.0
2024-03-24 16:25	0	0.0	0	0.0	0	0.0	0	0.0
2024-03-24 16:30	0	0.0	0	0.0	0	0.0	0	0.0
2024-03-24 16:35	1	0.25	0	0.0	1	0.25	0	0.0
2024-03-24 16:40	1	0.25	0	0.0	1	0.25	0	0.0
2024-03-24 16:45	1	0.25	0	0.0	1	0.25	0	0.0
2024-03-24 16:50	0	0.0	0	0.0	0	0.0	0	0.0

Exponential Smoothing and Random Forest models were trained on the time-series dataset. The models, which were trained on the training data, were utilized to generate forecasts for future table occupancy over a period equivalent to the size of the test set. The performance of the models was compared using common time-series error metrics: Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). The model comparison results are presented in Table 5. MAE measures the magnitude of prediction errors in the original units of the data, independent of their direction, making it easily interpretable. MSE disproportionately penalizes large errors (outlier predictions) by squaring them, which makes the model more sensitive to such deviations. While RMSE, like MSE, also emphasizes large errors, it brings the error metric back to the same scale as the original data by taking the square root, thus facilitating the interpretation of the error's magnitude. For these reasons, all three error metrics were utilized in the model comparison.

Table 5. *A Comparison of Model Performance.*

Table	**Model**	**MAE**	**MSE**	**RMSE**
Table 1	Random Forest	1.2417	2.5020	1.5818
	Exponential Smoothing	**1.2323**	**2.0275**	**1.4239**
Table 2	Random Forest	1.6021	3.4713	1.8631
	Exponential Smoothing	**1.3803**	**2.4817**	**1.5753**

Upon examining Table 5, it was observed that Exponential Smoothing, a classical statistical time-series model, outperformed Random Forest, a machine learning ensemble model, across both tables and all error metrics (MSE, RMSE, MAE). The lower error scores of the Exponential Smoothing model indicate that it forecasts future table occupancy levels with higher accuracy.

By its nature, restaurant table occupancy data contains strong trend and seasonality components. Exponential Smoothing is inherently designed to make forecasts based on weighted averages and these types of temporal patterns in historical data. Although Random Forest is a powerful model for capturing complex, non-linear relationships, its performance can be limited when the dataset is dominated by regular, periodic components like strong trends and seasonality. The Exponential Smoothing model, by its core algorithm, is specifically designed to model these trend and seasonality structures directly. Therefore, the nature of the problem falls within the primary domain of Exponential Smoothing, which demonstrates the model's ability to successfully capture the structural temporal dependencies in the data.

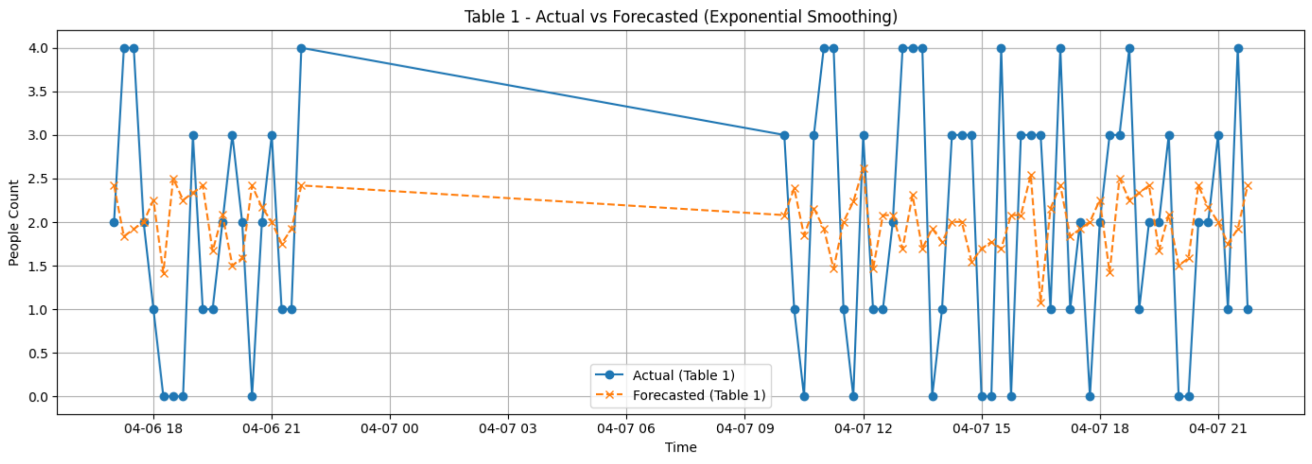

Figure 9. *A comparison between the actual occupancy and the values forecasted by the Exponential Smoothing model for first table of the restaurant.*

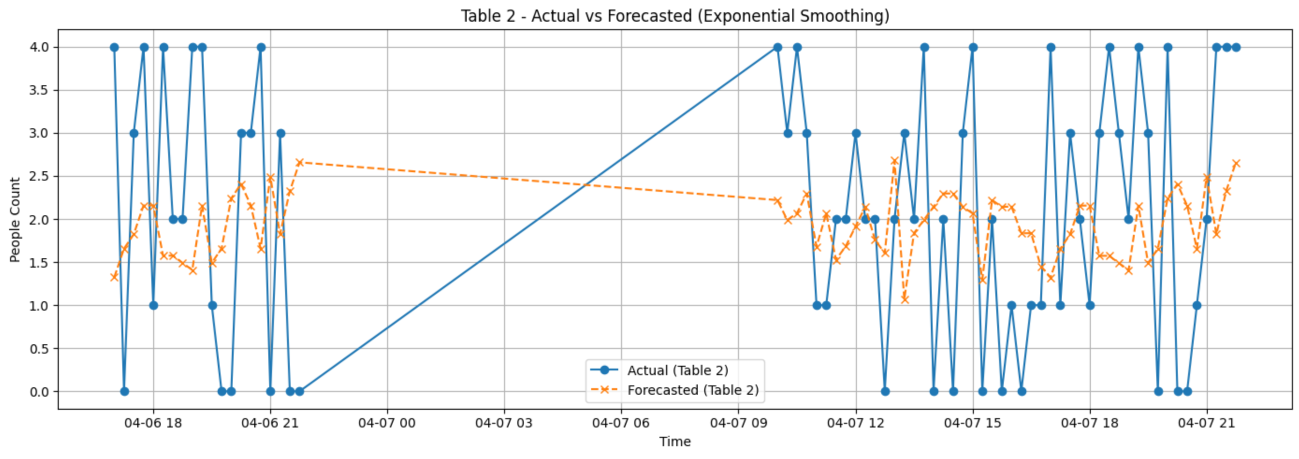

Figure 10. *A comparison between the actual occupancy and the values forecasted by the Exponential Smoothing model second table of the restaurant.*

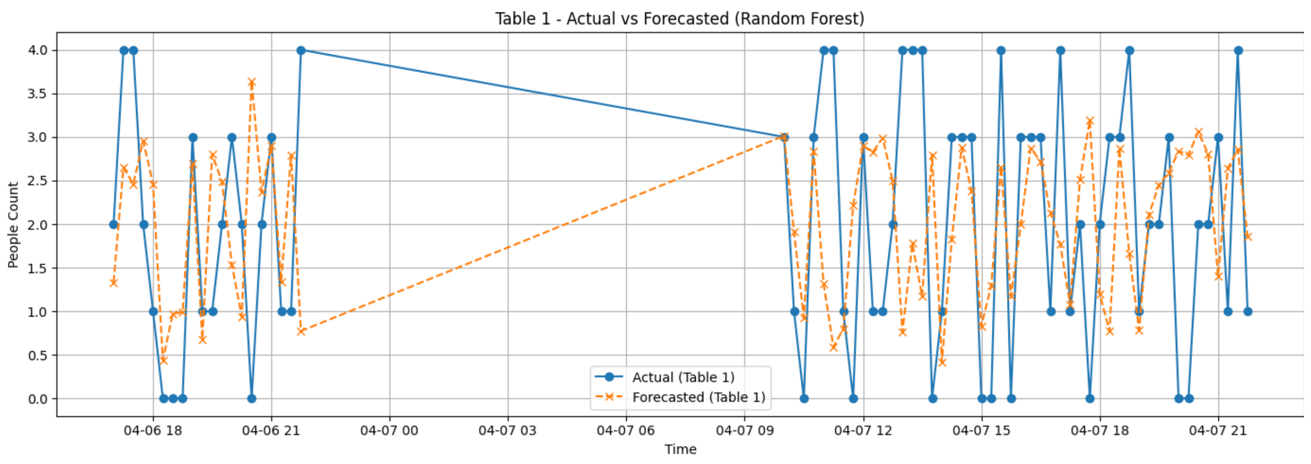

Figure 11. *A comparison between the actual occupancy and the values forecasted by the Random Forest model for first table of the restaurant.*

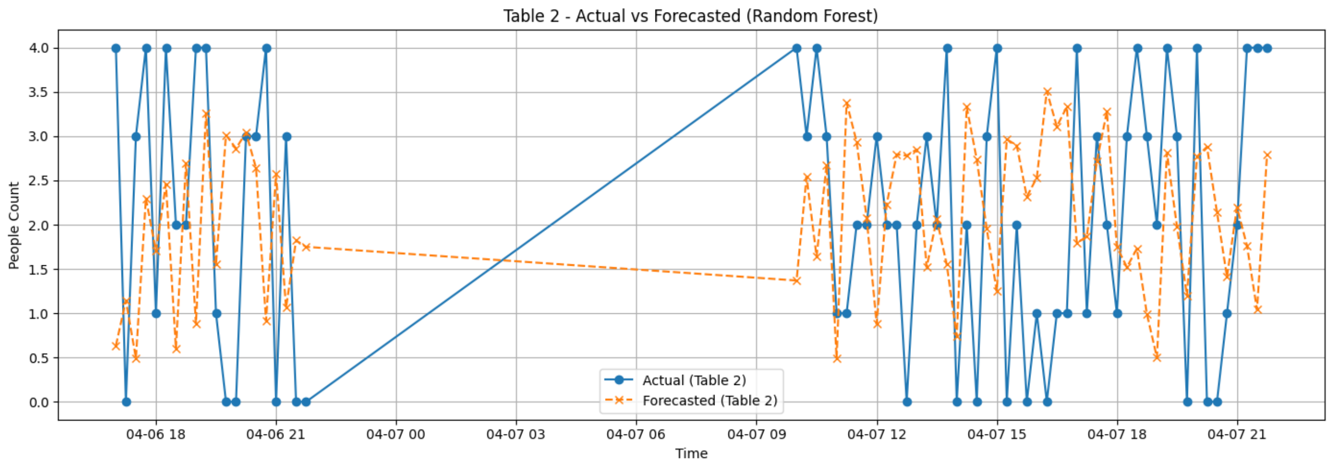

Figure 12. *A comparison between the actual occupancy and the values forecasted by the Random Forest model for second table of the restaurant.*

Figures 9-12 provide a visual analysis of the forecasting performance of the Exponential Smoothing and Random Forest models on the test data. Figures 9 and 11 show the forecasts for first table of the restaurant from the Exponential Smoothing and Random Forest models, respectively, while Figures 10 and 12 present the same comparison for second table of the restaurant. In these graphs, the solid blue line represents the actual table occupancy counts (Actual), while the dashed orange line represents the forecast values generated by the model (Forecasted). This visual inspection of the graphs corroborates the numerical error metrics (MSE, RMSE, MAE) presented in Table 5.

The forecasts from the Exponential Smoothing model exhibit a structure that follows the general trend and level of the actual data but smooths out sudden fluctuations. Instead of precisely mimicking the sharp jumps in the actual data (between zero and four people), the model follows a more stable and consistent path that reflects the average of these fluctuations. This behavior is a result of the Exponential Smoothing model's inherent nature, which involves taking weighted averages of past data to filter out noise and capture the underlying structure of the series. This approach prevents the model from making large errors, thereby reducing the overall error rate. In contrast, the forecasts from the Random Forest model are visibly more erratic and volatile. The model, as represented by the orange line, aggressively attempts to capture the peaks and troughs of the actual data, but in doing so, it often misses the precise timing or magnitude. The resulting spiky nature of the forecast graph visually confirms the model's lower overall stability, which in turn leads to higher error metrics.

Table 6. *Comparative Analysis of Occupancy Detection Studies in the Literature*.

Study (Author, Year)	Application Area	Data Set Used	Method/Techniques	Obtained Results
Bong et al. (2008)	Parking Lot Area	Custom-generated video images	Image Processing, Object Detection, Edge Detection	Parking lot occupancy detection with a 95% accuracy rate
Halgaš & Pirník (2015)	Parking Lot Area	Camera images	Image Processing, Video Detection	Average occupancy detection accuracy of 90%
Petersen et al. (2016)	Room Occupancy	Sensor and Image data	Image-based methods, Real-time verification	93% accuracy rate based on sensor data
Monti et al. (2022)	Classroom Occupancy	Camera images	Edge-based transfer learning, Deep Learning	91% accuracy in estimating in-class occupancy rate
Nguyen et al. (2015)	Seat Occupancy	IR sensors, Capacitive sensors	Real-time sensor data	97% accuracy in seat occupancy detection
Datenwissen (2023)	Smart Occupancy Tracking	YOLO, Real-Time Detection	Human density monitoring, capacity management, social distancing control	-
Traf-Sys (2023)	People Counting in Restaurants	Image Processing, YOLO	Peak hour detection, operational efficiency	-
This study	**Restaurant Table Occupancy**	**Custom-generated table image dataset**	**YOLOv8, Dynamic table and person detection, Time Series analysis**	**94% table occupancy detection accuracy in a restaurant environment, generation of time-series-based operational data**

A comparative analysis of occupancy detection studies in the literature is presented in Table 6. Based on the findings from these studies, the primary reasons for selecting the YOLO algorithm are its high accuracy rate, real-time performance, and its ability to adapt to dynamic environments. Furthermore, a clear need emerges for the creation of a specialized dataset for table occupancy and people counting in restaurant environments, moving beyond generic solutions. This study contributes to the literature by addressing these identified gaps through the development of restaurant-specific datasets and dynamic table detection approaches.

5. Conclusion and Future Works

In this study, an end-to-end intelligent system for real-time table occupancy detection and forecasting in a restaurant environment was developed and validated. By utilizing a custom-trained YOLOv8 model and integrating it with a time-series forecasting module, the system moves beyond simple monitoring to provide actionable data for operational planning. The findings indicate that this system can serve as a powerful tool for enhancing efficiency and customer satisfaction in the restaurant industry.

The primary advantages of this study include its ability to automatically detect real-time table occupancy with high accuracy in restaurant environments, which is achieved through the use of custom-created datasets and the YOLO algorithm. Furthermore, this study presents a complete pipeline, from real-time visual data capture to predictive analytics, rather than just a standalone detection model. Through the integration of the detection module with a forecasting module using Exponential Smoothing and Random Forest models, along with heatmap analyses, businesses can make effective decisions to increase operational efficiency and enhance customer satisfaction. The implementation of a dynamic table isolation algorithm using Euclidean distance ensures a reliable focus on the correct table, while automated data logging via a cron-scheduled API demonstrates a practical and scalable architecture suitable for real-world deployment.

However, this study has several limitations. First, since the developed models were trained on the specific camera angles and lighting conditions of one restaurant, they may require recalibration and retraining for different restaurant environments or camera placements. Second, the model's accuracy can be affected by environmental factors such as sudden changes in lighting, shifts in camera angles, or unexpected repositioning of tables and chairs. Third, the dataset was collected over a limited period in a single restaurant environment, which may limit the model's generalizability compared to environments with greater data diversity. Finally, the occupancy forecasting module relies solely on historical occupancy data. The absence of external variables—such as weather conditions, local events, holidays, or active marketing promotions—may cap the module's performance at a certain level.

Future work could enhance the model's generalizability by using larger and more diverse datasets collected from various restaurant types and under different environmental conditions. Transfer learning approaches could be applied to adapt the table occupancy detection model for different restaurant chains or customer profiles. Additionally, the forecasting module could be improved by incorporating external variables, such as weather conditions, local events, holidays, or active marketing promotions, rather than relying solely on historical data.

Declaration of Interest

The authors declare that there is no conflict of interest.

Acknowledgements

We express our thanks to Protel A.Ş. for providing the test environment used in this study and contributing to the acquisition of the data used in the research.

References

[1] B. Esposito, M. R. Sessa, D. Sica, and O. Malandrino, "Service innovation in the restaurant sector during COVID-19: Digital technologies to reduce customers' risk perception," The TQM Journal, vol. 34, no. 7, pp. 134–164, 2022.

[2] M. E. Rodríguez-López, J. M. Alcántara-Pilar, S. Del Barrio-García, and F. Muñoz-Leiva, "A review of restaurant research in the last two decades: A bibliometric analysis," International Journal of Hospitality Management, vol. 87, p. 102387, 2020.

[3] D. Marr and T. Poggio, "A computational theory of human stereo vision," Proceedings of the Royal Society of London. Series B. Biological Sciences, vol. 204, no. 1156, pp. 301–328, 1979.

[4] J. Canny, "A computational approach to edge detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 6, pp. 679–698, 1986.

[5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, 2004.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2014, pp. 580–587.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," Advances in Neural Information Processing Systems, vol. 28, 2015.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Advances in Neural Information Processing Systems, vol. 25, 2012.

[9] J. Redmon, "You only look once: Unified, real-time object detection," in Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2016.

[10] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2017, pp. 7263–7271.

[11] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," arXiv preprint, arXiv:2004.10934, 2020.

[12] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics. [Accessed: Aug. 13, 2024].

[13] D. B. L. Bong, K. C. Ting, and K. C. Lai, "Integrated approach in the design of car park occupancy information system (COINS)," IAENG International Journal of Computer Science, vol. 35, no. 1, 2008.

[14] J. Halgaš and R. Pirník, "Monitoring of parking lot traffic using a video detection," Acta Technica Corviniensis - Bulletin of Engineering, vol. 8, no. 3, pp. 17–20, 2015.

[15] K. Pannerselvam, "Adaptive parking slot occupancy detection using vision transformer and LLIE," in 2021 IEEE International Smart Cities Conference (ISC2), 2021.

[16] M. Kročka, P. Dakić, and V. Vranić, "Extending parking occupancy detection model for night lighting and snowy weather conditions," in 2022 IEEE Zooming Innovation in Consumer Technologies Conference (ZINC), 2022.

[17] J. D. Ellis, S. Sam, and H. A. Smit, "An analysis of lightweight convolutional neural networks for parking space occupancy detection," in 2021 IEEE International Symposium on Multimedia (ISM), 2021.

[18] S. Goumiri, D. Benboudjema, and W. Pieczynski, "One convolutional layer model for parking occupancy detection," in 2021 IEEE International Smart Cities Conference (ISC2), 2021.

[19] S. Funck, N. Mohler, and W. Oertel, "Determining car-park occupancy from single images," in Proc. IEEE Intelligent Vehicles Symposium, 2004.

[20] S. Petersen, M. Sønderskov, H. Rasmussen, and J. Thomsen, "Establishing an image-based ground truth for validation of sensor data-based room occupancy detection," Energy and Buildings, vol. 130, pp. 787–793, 2016.

[21] L. Monti, C. Bianchi, A. Venturini, and M. Capra, "Edge-based transfer learning for classroom occupancy detection in a smart campus context," Sensors, vol. 22, no. 10, p. 3692, 2022.

[22] H. H. Nguyen, J. Li, and W. Ng, "Real-time detection of seat occupancy & hogging," in Proc. 2015 International Workshop on Internet of Things Towards Applications, 2015.

[23] Datenwissen. (2023). Smart Occupancy Monitoring with Computer Vision. Available online: https://datenwissen.com/nwarch-ai/occupancy-monitoring/. Accessed: June 23, 2025.

[24] Traf-Sys. (2023). How Restaurants Can Use People Counting Systems. Available online: https://www.trafsys.com/how-restaurants-can-use-people-counting-systems/. Accessed: June 23, 2025.

[25] G. van Rossum and F. L. Drake, Python 3 Reference Manual, Scotts Valley, CA, USA: CreateSpace, 2009. [Online]. Available: http://www.python.org.. Accessed: June 23, 2025

[26] G. Bradski, “The OpenCV library,” Dr. Dobb's Journal of Software Tools, 2000.

[27] D. Tzutalin, LabelImg. Git code, 2015. [Online]. Available: https://github.com/HumanSignal/labelImg.

[28] W. Rawat and Z. Wang, “Deep convolutional neural networks for image classification: A comprehensive review,” Neural Computation, vol. 29, no. 9, pp. 2352–2449, 2017.

[29] J. Terven, D. M. Córdova-Esparza, and J. A. Romero-González, “A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS,” Machine Learning and Knowledge Extraction, vol. 5, no. 4, pp. 1680–1716, 2023.

[30] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, and C. L. Zitnick, “Microsoft COCO: Common objects in context,” in Computer Vision – ECCV 2014: Proceedings of the 13th European Conference on Computer Vision, Zurich, Switzerland, Sept. 6–12, 2014, Part V, pp. 740–755, Springer International Publishing.

[31] M. A. Schwarz, “Linux job scheduling,” Linux Journal, vol. 2000, no. 77es, p. 8-es, 2000.

[32] J. Flask documentation, Flask. [Online]. Available: https://flask.palletsprojects.com/en/stable/. [Accessed: Dec. 29, 2024].

[33] E. S. Gardner Jr, “Exponential Smoothing: The State of the Art—Part II,” International Journal of Forecasting, vol. 22, no. 4, pp. 637–666, 2006.

[34] S. J. Rigatti, “Random Forest,” Journal of Insurance Medicine, vol. 47, no. 1, pp. 31–39, 2017.