Pallet Detection using Image Processing

István Ferenc Farkas*, Sándor Szénási†
John von Neumann Faculty of Informatics, Obuda University, Budapest, Hungary
*farkas.istvan@stud.uni-obuda.hu, †szenasi.sandor@nik.uni-obuda.hu
Faculty of Economics and Informatics, J. Selye University, Komárno, Slovakia
†szenasis@ujs.sk

2025 IEEE 29th International Conference on Intelligent Engineering Systems (INES) | 979-8-3315-9771-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/INES67149.2025.11078191

***Abstract*—This paper presents a software capable of real-time pallet detection and positioning based on live camera feeds. Automated pallet handling plays an increasingly vital role in logistics and industrial processes, significantly enhancing efficiency and reducing workplace risks. To develop the software, computer vision and deep learning techniques were utilized, specifically the YOLOv8 model. The training was conducted using approximately 4,000 images, resulting in high detection accuracy. The system can determine the location and type of pallets and can be integrated into various industrial applications, such as automated material handling systems. Pallet identification is performed using QR codes, and the software employs a MongoDB database to store data associated with the identified pallets. Based on the results, the software can serve as an effective tool for optimizing warehouse and logistics systems, particularly in environments where automation is critical.**

***Index Terms*—pallet detection, logistics, automation, machine learning, computer vision**

I. Introduction

Pallets are ubiquitous structures essential for shipping and industry, made of wood or plastic. They serve various purposes, from simple garden tables to crucial components in complex global trade logistics. The increasing role of automation in pallet handling is becoming prominent in the industry for several important reasons. Automated systems improve efficiency and productivity by being faster and more accurate than human beings. They also enhance workplace safety by reducing the risk of injuries associated with manual pallet handling. Furthermore, automated pallet handling optimizes warehouse operations by enabling quicker and more precise pallet movement. There are also environmental benefits, as these systems consume less energy and produce less waste than traditional methods.

This research aims to develop software capable of real-time pallet recognition and positioning using live camera images. The key aspects of this project include creating a system for automatic detection and localization of pallets in camera view, utilizing artificial intelligence and machine vision methods for image processing and analysis, and enabling precise identification of pallet positions. This software could significantly benefit industrial processes by enhancing the efficiency of automated material handling systems, improving route planning and navigation for Automated Guided Vehicles (AGVs), and aiding in inventory management through automatic pallet identification [1]. The goal is to create software that improves the efficiency and safety of various industrial applications through accurate pallet recognition and localization based on live camera images.

II. Related Work

A. Nvidia

NVIDIA developers have created a highly accurate pallet recognition system using live camera images [2]. The system employs Synthetic Data Generation (SDG) to train the neural network, generating 2D and 3D computer-created objects instead of using real images. The development process integrated OpenUSD (Universal Scene Description) and synthetic data generation, enabling the creation and management of complex 3D scenes. This approach allows the production of accurate, detailed, and diverse datasets for model training, ensuring efficiency and high recognition accuracy.

Synthetic data generation is advantageous when collecting real data is challenging or expensive [3], [4]. It allows developers to create unlimited, varied, and accurate datasets to improve model performance. The combination of OpenUSD and synthetic data generation represents a significant advancement in Computer Vision (CV) and Machine Learning (ML) applications. This method not only improves model accuracy but also allows developers to use unlimited and diverse data for training deep learning algorithms, resulting in a system capable of recognizing and detecting even stacked pallets.

B. Based on point cloud data

This study [5] presents a new pallet recognition method for AGVs in warehouse environments. Based on point cloud data collected by a 3D sensor, the method offers faster and more reliable recognition than traditional approaches. It consists of five main modules:

- Point cloud preprocessing: Removes noise and unnecessary information.
- Key point selection: Identifies characteristic points crucial for recognition and matching.
- Feature description: Uses Adaptive Color Fast Point Feature Histogram (ACFPFH) to combine color and geometric data.
- Surface matching: Creates and refines matches between the pallet pattern and the current environment.
- Point cloud registration: Utilizes RANSAC and ICP algorithms [6] for reliable alignment between the pallet and AGV.

The proposed method significantly accelerates the registration process and improves recognition accuracy. It performs well in various warehouse environments, including between shelves and under varying lighting conditions. This enhances AGV performance, increasing warehouse system efficiency and reducing accident risks.

C. Machine Learning Frameworks

YOLO is a single-pass object detection model [7], meaning it can detect objects in a single pass. As a result, it is extremely fast and efficient, making it ideal for real-time applications [8]. YOLO operates by dividing the input image into a grid of cells. The model predicts a bounding box and a set of class probabilities for each cell. The class probabilities represent the likelihood that a particular class is present within the cell. In cases where an object is detected multiple times, non-maximum suppression is used to remove duplicates, thereby improving accuracy.

ONNX is an open-source format [9] and ecosystem that enables easy transfer and interoperability of machine learning models between different frameworks. It breaks down barriers between machine learning models and frameworks and allows model transfer between different frameworks (e.g., PyTorch to TensorFlow). It is widely used in image recognition, speech recognition, natural language processing, and robotics.

GPU acceleration is becoming increasingly common for mathematical and data processing tasks, especially deep learning algorithms. This is because GPUs are particularly well-suited for parallel processing large amounts of data compared to CPUs, which is highly advantageous when simultaneously processing many independent data points. In deep learning, the rapid processing of large amounts of data required for network training is critically important, and GPU-based acceleration significantly contributes to increasing efficiency. Two leading platforms are available for GPU-based deep learning tasks: CUDA and ROCm [10]. CUDA is a framework developed and supported by NVIDIA, and it integrates closely with the hardware and software systems of NVIDIA GPUs. This means that using CUDA often provides the highest possible performance on NVIDIA hardware, as the framework is optimized explicitly for these architectures. ROCm, on the other hand, is an open-source framework developed and supported by AMD, primarily optimized for AMD GPUs. Although ROCm supports high performance on AMD GPUs. Unfortunately, only a few AMD products currently support this solution, which may limit its adoption and applicability.

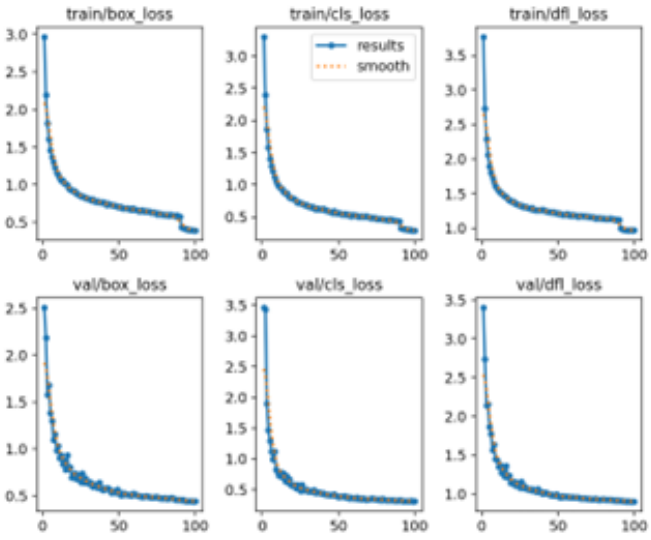

Fig. 1. Loss bounding box

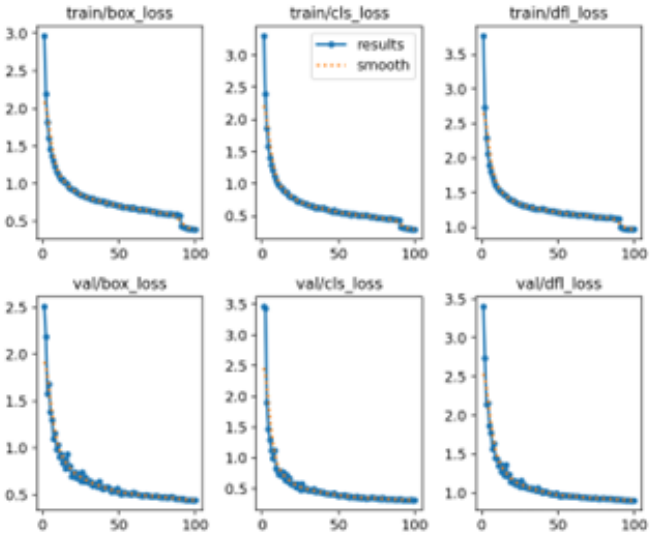

Fig. 2. Loss segmentation

III. Methodology

A. Loss function

Loss in machine learning is a metric that explains how much our model errs in its predictions for a given instance. The loss value is a number that shows how much the predicted result deviates from the true result. If our model performs perfectly, meaning all its predictions exactly match reality, then the loss value will be zero. This means that our model could predict every data point accurately, and no correction is needed. However, if our model makes mistakes, i.e., its prediction does not match the actual data, then the loss value increases. The higher the loss value, the greater the error, and the more the predictions deviate from the real data.

The types of loss can vary depending on the problem to solve. Mean Squared Error (MSE) is a commonly used loss function for regression problems, while Cross-Entropy Loss is common for classification problems. Every loss function aims to minimize the error in the model and increase its accuracy. The optimization of the loss function occurs during model training, where the model's parameters are adjusted to minimize the loss value.

B. Comparing the two training

Bounding boxes and segmentation are two different methods in object recognition, each with its own advantages and disadvantages. The results of the first two trainings are shown in Fig. 1 and Fig. 2.

The bounding boxes method, used for the first training, involves enclosing objects in a rectangular box that surrounds the object. The box coordinates (*x*, *y*, *width*, *height*) determine the location and size of the object in the image. The advantages of this method include simplicity and efficiency. It's easy to implement and quick to compute, making it suitable for real-time applications as it requires less computational resources. It also works well for various object types as it doesn't require detailed information about the shape of the object. However, the bounding boxes method is not always accurate, especially for complex or irregularly shaped objects, and doesn't provide information about the exact boundaries of the object, only a rough outline. Also, it doesn't allow for determining the orientation of the pallet, as visible in Fig. 3.

Segmentation aims to precisely separate objects from the background at the pixel level, which is the method applied in the second training. The advantages of segmentation include high accuracy and detail, as it determines object boundaries and areas at the pixel level. This is particularly useful for complex or irregularly shaped objects. However, segmentation requires more computational resources, which can result in slower processing and is more complex to implement and fine-tune. Using this method, it's possible to determine the orientation of the pallet in further steps (Fig. 4).

With both methods, it is possible to achieve exceptionally high accuracy, numerically exceeding 85%. The results achieved with the first method, bounding boxes, provided fast and efficient object recognition, but the second method, segmentation, offers even greater detail and precision. As a result, it's possible to determine the position and orientation of objects, especially pallets, much more accurately. Since the precise location and orientation of pallets are crucial for further processing steps, it is necessary to continue development using segmentation, as this method significantly improves the system's reliability and precision.

Fig. 3. Bounding box

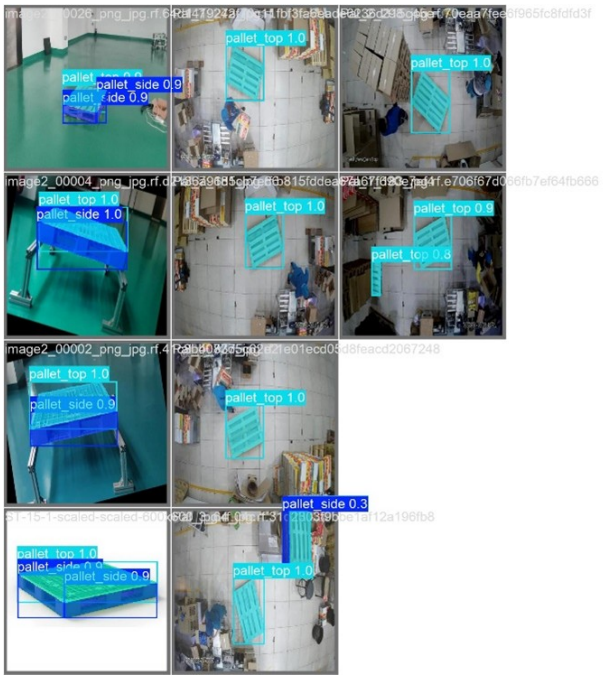

Fig. 4. Segmentation

C. Positioning

If a pallet side is detected, subsequent calculations determine the position of the pallet in the image and whether it is centered relative to two fixed points. If the pallet is indeed centered, the guidelines turn green, indicating correct placement. If not, the lines remain red, warning the user that the pallet isn't centered.

1) Pallet Side Detection: The system first detects the pallet side using the YOLOv8 model. Pallet detection is based on object classes, ensuring only the pallet side is observed during detection. Once the pallet side is recognized, the system converts the contours and position of the pallet into spatial coordinates to determine its location in the image.

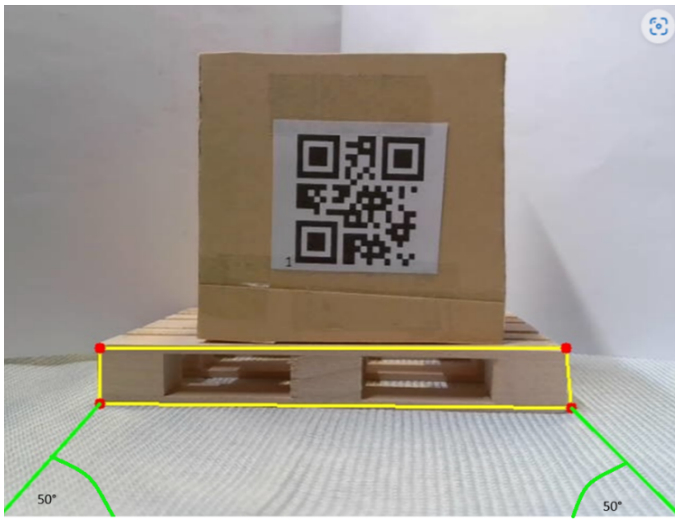

Fig. 5. Positioning

2) Calculating Pallet Position: After detecting the pallet side, the next step is calculating its position. Two fixed points are defined in the code as reference points to determine the position of the pallet. A line is drawn from each fixed point to the two lower corners of the pallet.

Examining if the pallet is centered involves measuring the difference in length between the lines drawn from the two fixed points. If the two lines are equal in length and form equal angles, the placement of the pallet is correct and centered. If the difference between the two lines is too large, it indicates the pallet isn't centered. Tolerance for angle and length can be defined in the code (Fig. 5).

3) Changing Guide Line Colors: The color change of the guidelines depends on how closely the two-line lengths match. If there's no difference in line lengths, the lines turn green, indicating the pallet is centered. If there's a difference, they turn red, warning the user that the pallet isn't centered. This visual cue helps the user identify when the pallet is properly positioned and when its position needs correction.

4) Final Visualization: The guidelines appear as clear visual tools on the screen. Green lines indicate that the pallet is in the correct position, while red lines show the pallet's deviation and warn the user that the pallet's positioning needs adjustment. This visual feedback helps the system make quick and accurate decisions regarding pallet placement, increasing workflow efficiency and reducing the possibility of human errors.

IV. Evaluation

The tests were conducted using a consumer PC, which allowed for the achievement of real-time processing speed. Based on the results, the system showed outstanding speed, capable of processing up to 30 frames per second.

In the test simulation, the system followed exactly the process it would perform in an industrial environment: continuously processing the camera image to detect pallets, identifying them based on QR codes, and updating the detection results on the interface in real time. The YOLOv8 model's deep learning algorithm quickly and accurately performed pallet recognition and guide line positioning.

The system's main performance indicators:

- FPS (frames per second): The processing speed consistently reached 30 FPS, which is excellent for real-time application.
- Accurate detection: The system was able to correctly identify pallets at different angles and lighting conditions, with an average detection rate of 89
- Fast response time: The live camera image on the user interface was continuously updated, and the data of detected pallets and QR codes appeared immediately (0.5 seconds).

Although the system testing was successful in virtual conditions, precise industrial applicability requires further investigation. Testing in an industrial environment is important because factors present in warehouses and production lines - such as greater distances, varying lighting, moving equipment, and noisy environments - can significantly affect system performance.

Based on these test results, it can be said that the system performed excellently in the test environment and is suitable for real-time pallet recognition. The achieved processing speed of 30 frames per second is an outstanding result that guarantees fast system operation. The next steps should focus on testing in an industrial environment to ensure performance and reliability in real applications.

The limitations of the testing environment compared to industrial environments are the following:

- Shorter distances: The camera was relatively close to the pallets, which made QR code detection easier.
- Stable environment: There were no moving objects or major disruptive factors that could have reduced detection accuracy.
- Constant lighting: Lighting was constant, while in industrial environments, lighting can vary due to shadows or reflections.

V. Conclusions

This paper presented the development of a real-time pallet recognition and positioning solution using machine vision and deep learning algorithms. As automation plays an increasingly important role in industrial and logistics processes, developing an efficient, accurate, and user-friendly solution is essential in advancing modern systems.

Position and identification are two key factors in pallet handling, and automating these can save significant time and costs. When selecting the necessary tools and technologies, the YOLOv8 deep learning model has been chosen, which performs exceptionally well in real-time object recognition tasks.

The system consists of three main parts: object recognition, QR code-based identification, and data management. Object recognition is performed using the YOLOv8 model, which is

trained specifically for pallet identification. For this purpose, a database of about 4,000 images depicting pallets in various situations and angles was created. The images are annotated manually to obtain accurate and relevant results during model training.

QR codes are responsible for pallet identification. Each QR code placed on the pallet or cargo contains a unique identifier for that pallet, which the system uses to query associated data. This data is stored in a NoSQL database, MongoDB. This database server is excellent for handling dynamically changing data as it doesn't require a predefined data structure and is easily scalable.

At the bottom of the interface, a list of detected pallets and detailed data extracted from QR codes, such as pallet type, size, or position in the warehouse, appear. If a pallet is centered relative to the reference determined by the guidelines, the lines turn green. If the pallet's position is incorrect, the lines remain red, alerting the user to the need for correction. The final system can be applied in numerous industrial areas, especially in warehousing, where quick and accurate identification of pallets is essential. Additionally, the system can be expanded to recognize other types of objects, further increasing its potential applications.

ACKNOWLEDGMENT

The authors would like to thank the High-Performance Computing Research Group of Óbuda University for its valuable support. The authors would like to thank NVIDIA Corporation for providing graphics hardware for the experiments.

REFERENCES

[1] H.-W. Cheong and H. Lee, "Concept design of agv (automated guided vehicle) based on image detection and positioning," *Procedia computer science*, vol. 139, pp. 104–107, 2018.

[2] J. Welsh, "Developing a pallet detection model using openusd and synthetic data." [Online]. Available: https://developer.nvidia.com/blog/developing-a-pallet-detection-model-using-openusd-and-synthetic-data

[3] A. I. Károly and P. Galambos, "Automated dataset generation with blender for deep learning-based object segmentation," in *2022 IEEE 20th Jubilee World Symposium on Applied Machine Intelligence and Informatics (SAMI)*. IEEE, 2022, pp. 000 329–000 334.

[4] V. Patakvölgyi, L. Kovács, and D. A. Drexler, "Synthetic data generation based on microscopic images of cancer cells," in *2024 IEEE 24th International Symposium on Computational Intelligence and Informatics (CINTI)*. IEEE, 2024, pp. 209–214.

[5] Y. Shao, Z. Fan, B. Zhu, M. Zhou, Z. Chen, and J. Lu, "A novel pallet detection method for automated guided vehicles based on point cloud data," *Sensors*, vol. 22, no. 20, p. 8019, 2022.

[6] X. Huang and M. Hu, "3d reconstruction based on model registration using ransac-icp algorithm," in *Transactions on edutainment XI*. Springer, 2015, pp. 46–51.

[7] G. Jocher, A. Stoken, J. Borovec, L. Changyu, A. Hogan, A. Chaurasia, L. Diaconu, F. Ingham, A. Colmagro, H. Ye *et al.*, "ultralytics/yolov5: v4. 0-nn. silu () activations, weights & biases logging, pytorch hub integration," *Zenodo*, 2021.

[8] V. Tadic, A. Odry, Z. Vizvari, Z. Kiraly, I. Felde, and P. Odry, "Electric vehicle charging socket detection using yolov8s model," *Acta Polytechnica Hungarica*, vol. 21, no. 10, 2024.

[9] "Onnx documentation." [Online]. Available: https://onnx.ai/onnx

[10] N. Kondratyuk, V. Nikolskiy, D. Pavlov, and V. Stegailov, "Gpu-accelerated molecular dynamics: State-of-art software performance and porting from nvidia cuda to amd hip," *The International Journal of High Performance Computing Applications*, vol. 35, no. 4, pp. 312–324, 2021.