algorithms

MDPI

Article

Computer Vision Algorithms on a Raspberry Pi 4 for Automated Depalletizing

Danilo Greco [1], Majid Fasihiany [2], Ali Varasteh Ranjbar [3], Francesco Masulli [2,3,*], Stefano Rovetta [2,3] and Alberto Cabri [2,4]

[1] Department of Management, Economics and Industrial Engineering, Politecnico di Milano, 20156 Milan, Italy
[2] Vega Research Laboratories s.r.l., 16121 Genoa, Italy
[3] Department of Informatics, Bioengineering, Robotics and Systems Engineering, Università degli Studi di Genova, 16146 Genoa, Italy
[4] Department of Informatics "Giovanni degli Antoni", Università degli Studi Statale di Milano, 20122 Milan, Italy
* Correspondence: francesco.masulli@unige.it

Abstract: The primary objective of a depalletizing system is to automate the process of detecting and locating specific variable-shaped objects on a pallet, allowing a robotic system to accurately unstack them. Although many solutions exist for the problem in industrial and manufacturing settings, the application to small-scale scenarios such as retail vending machines and small warehouses has not received much attention so far. This paper presents a comparative analysis of four different computer vision algorithms for the depalletizing task, implemented on a Raspberry Pi 4, a very popular single-board computer with low computer power suitable for the IoT and edge computing. The algorithms evaluated include the following: pattern matching, scale-invariant feature transform, Oriented FAST and Rotated BRIEF, and Haar cascade classifier. Each technique is described and their implementations are outlined. Their evaluation is performed on the task of box detection and localization in the test images to assess their suitability in a depalletizing system. The performance of the algorithms is given in terms of accuracy, robustness to variability, computational speed, detection sensitivity, and resource consumption. The results reveal the strengths and limitations of each algorithm, providing valuable insights for selecting the most appropriate technique based on the specific requirements of a depalletizing system.

Keywords: depalletizing systems; single-board computer; Raspberry Pi 4; computer vision; object detection; pattern matching; scale-invariant feature transform (SIFT); Oriented FAST and Rotated BRIEF (ORB); Haar cascade classifier; industrial automation; robotic manipulators

check for updates

Citation: Greco, D.; Fasihiany, M.; Ranjbar, A.V.; Masulli, F.; Rovetta, S.; Cabri, A. Computer Vision Algorithms on a Raspberry Pi 4 for Automated Depalletizing. *Algorithms* **2024**, *17*, 363. https://doi.org/10.3390/a17080363

Academic Editor: Guanqiu Qi

Received: 21 June 2024
Revised: 14 August 2024
Accepted: 15 August 2024
Published: 18 August 2024

1. Introduction

In modern factory settings, depalletizing, which involves the ordered removal of specific objects from a pallet, is a crucial task for maintaining efficient material handling and inventory management. Traditional depalletizing methods are labor-intensive, time-consuming, and prone to errors, leading to potential safety risks and productivity losses. Consequently, there is a growing demand for automated depalletizing systems that can streamline this process, reduce human intervention, and enhance overall operational efficiency. Automated depalletizing systems equipped with robotic arms can operate continuously without fatigue, significantly increasing the throughput compared to manual depalletizing. Robots can handle repetitive tasks with consistent speed and precision, leading to higher productivity and reduced cycle times. For industrial use cases, several solutions have been available for quite a long time. Many of them rely heavily on computer vision algorithms for accurate object detection and localization. They leverage image processing algorithms to identify and locate the desired objects on the pallet, enabling a robotic system to pick and move them precisely.

With technical, economical, and societal changes, however, automation is becoming more and more pervasive. Robotics has entered small productive contexts and even everyday life, and machine learning-based artificial intelligence is being adopted in almost every aspect of productivity. In this scenario, the need has arisen for solving the depalletizing problem in a cost-effective way so as to make the technology available, for instance, to retailers of goods such as wood pellet bags or to small warehouses, just to mention two use cases.

Out of a complete depalletizing system, involving mechanical parts, power, and other components, this paper studies the sub-problem of identifying and locating objects on a pallet. These are assumed to have mildly irregular shapes and to be featuring possibly complex textures and patterns. The focus is on small-scale, cost-effective computers suitable for implementing edge-computing systems, i.e., systems not relying on either the cloud or a central server. The reference architecture here is a Raspberry PI 4, a single-board computer built around a custom Broadcom SoC particularly suitable for Internet of Things (IoT) and edge-computing applications due to its low cost, low power consumption, and ease of interfacing with cameras and other sensors, even if these advantages are counterbalanced by its small memory size and low computing power.

It should be noted, however, that working under the hypothesis of computing limitations constitutes a specific design requirement. These limitations rule out the use of more sophisticated, but computationally demanding, solutions. The focus is therefore on some classic object detection algorithms that, while lacking the flexibility of recent deep learning-based models, can be easily implemented in this kind of hardware architecture.

This paper presents a comprehensive comparative analysis of four computer vision algorithms for implementing a depalletizing system as follows: pattern matching, scale-invariant feature transform (SIFT), Oriented FAST and Rotated BRIEF (ORB), and Haar cascade classifier [1–8]. Each of these algorithms offers unique strengths and limitations, and their performance is evaluated based on several key factors, including accuracy, robustness to variability, computer speed, detection sensitivity, and resource consumption. Although in the literature, the four chosen algorithms have already been implemented on various Raspberry architectures [9–14], in this work, we developed them on a specific Raspberry PI 4 platform and evaluated their performance on the same test images specific to the depalletizing application so that we could compare them and choose the most suitable ones for a specific system, considering their accuracy and recognition latency.

The experimental setup is based on a Raspberry PI 4, a camera, and a simulated physical environment. In the following sections, the implementation of each technique is described in detail, and the results are thoroughly analyzed and discussed.

The paper organization is as follows: in Section 2, we provide a foundation for understanding the current state of the art. Section 3 details the experimental setup and the specific computer vision algorithms employed, including SIFT, ORB, and others, along with the criteria for their selection. Section 4 presents the findings from the experiments, comparing the performance of different algorithms based on indices such as accuracy, processing time, and resource consumption. This is followed by a discussion that interprets the results, highlighting the strengths and limitations of each algorithm and their implications for practical deployment in industrial settings. The paper ends with Section 5 summarizing the key insights and suggesting directions for future research.

2. Background and Motivation

2.1. Depalletizing Systems in Industrial Environments

A pallet is a structural element used to provide stability to a set of items when they are moved as a unit by mechanical means, such as a forklift. Typical pallets are wooden frames, but other materials are also used. There exist dimensional standards to facilitate handling at different locations (source, transportation, destination).

Depalletizing systems play a crucial role in modern industrial environments, where efficient material handling and inventory management are vital for streamlining operations and minimizing downtime [15–17]. These systems are designed to automate the process of either moving entire pallets or removing specific objects from a pallet, reducing the need for manual labour and mitigating potential safety risks associated with repetitive tasks.

Manual depalletizing can be labour-intensive, time-consuming, and prone to errors. Additionally, it may expose workers to potential hazards, such as repetitive strain injuries and accidents caused by the improper handling or lifting of heavy objects. Automated methods can employ single-purpose depalletizer machines, working on regular arrangements (see e.g., [18] for a description), or robotic arms. The latter, in turn, can be either programmed (e.g., [19]) or semi-autonomous, based on machine vision (e.g., [20]).

Focusing on machine vision-guided robotic arms, they have been one of the first pioneering experiments in AI-based robotics [21,22] and are currently widely available for traditional applications, as well as being the object of much current research [23–28].

However, when targeting resource- and cost-constrained computing as required in such environments as retail delivery automation or small-scale production, the options become more limited.

2.2. *Computer Vision Techniques for Object Detection and Localization*

Although the palletizing process aims at providing a regular arrangement of items, not all use cases for unpalletizing systems can rely on strict geometric assumptions. Figure 1 shows some examples of more complex cases.

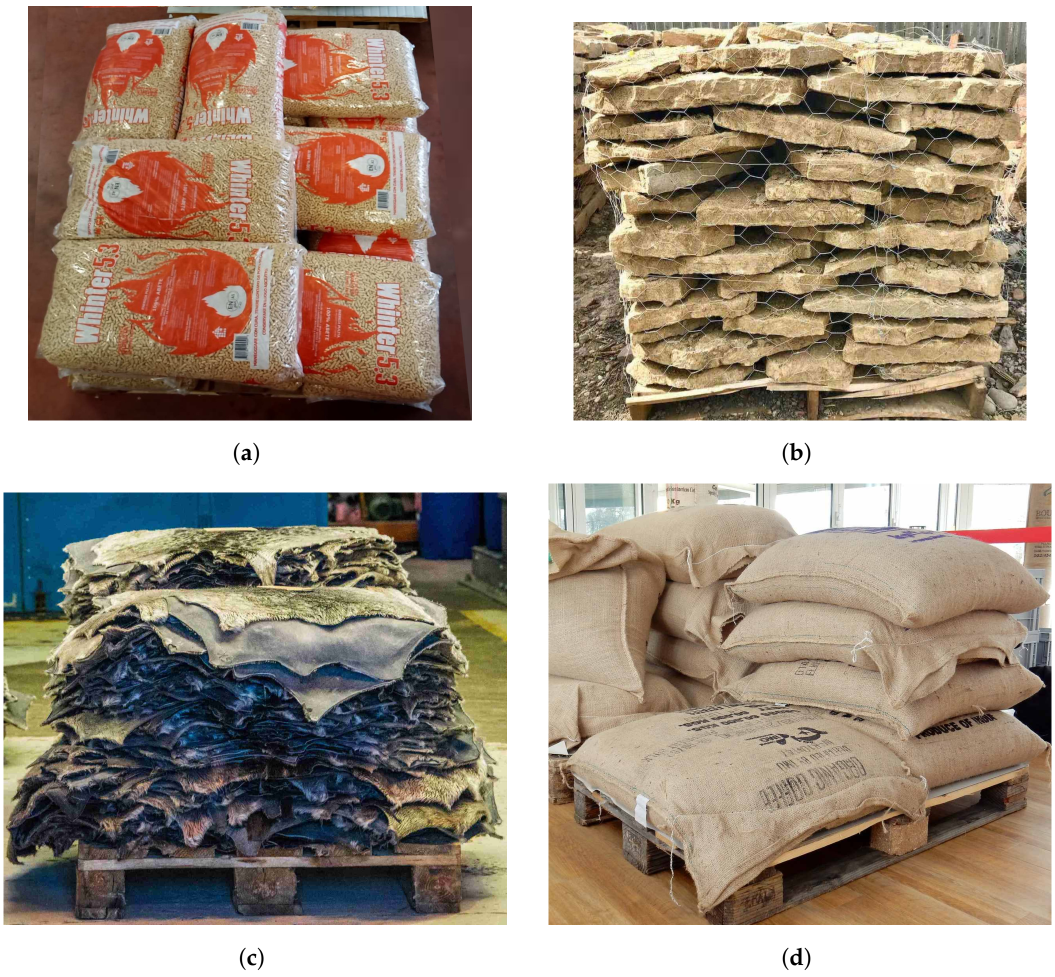

Figure 1. Some use cases: (**a**) pellet fuel bags, (**b**) stones, (**c**) leather (image credit: Ted McGrath on Flickr, CC BY-NC-SA 2.0), (**d**) coffee bags.

Computer vision techniques have revolutionized the field of object detection and localization, with numerous algorithms and approaches being developed and applied across various domains [29,30]. These techniques leverage sophisticated image processing algorithms and machine learning models to identify and locate specific objects within complex visual scenes [31].

2.3. Previous Work on Comparing Methods

Numerous studies have been conducted in the field of computer vision for object detection and localization, exploring various techniques and their applications in different domains. However, research specifically focused on the comparative analyses of these algorithms for depalletizing system implementation under resource constraints is relatively limited. The applications related to vision-based depalletizing reveal an abundance of scientific literature; however, the number of methods covered by each study may be restricted and is sometimes out of date. For instance, pioneering work was conducted in object detection with SIFT-based clustering [2] to pick and place objects; while this is presented as a single technique, it is highly flexible for different objects. In another work, Ahaitouf and Mansouri [3] propose two feature selection algorithms Haar-like feature selection and Local Binary Patterns (LPB) for the detection of a single object and multiple objects in the same scene and for both standard platforms and embedded systems. Bansal and Kumar [32] presented the performance of various object recognition approaches in a comparative analysis of SIFT, SURF, and ORB feature descriptors and multiple combinations of these feature descriptors. This experimental work was conducted using a public dataset, namely Caltech-101.

In other research [5], the development and comparison of two distinct approaches, a machine learning strategy based on Support Vector Machines (SVMs) [33] and a deep learning approach employing the YOLO (You Only Look Once) [34] network, for small target detection is analysed. Due to the restrictions in the region of interest (ROI) selection, the SVM-based method performs better in terms of computing resources and time consumption but suffers from accuracy and robustness problems in some cases, such as occlusion, illumination fluctuations, and tilting. Conversely, the YOLO network-based deep learning approach has better accuracy but has trouble in reaching real-time performance, particularly on onboard computers that have weight and power limitations.

Recent changes in the YOLO network architecture made it possible to run the system on low computing power edge devices, although the issue of detection latency may not be suitable for real-time operation. Furthermore, the training of deep learning algorithms requires a large amount of data, and the process of creating a dataset and manually labelling it takes a long time. As a result, object detection techniques should be properly chosen based on the application and design requirements, and adjustments must be made as needed. For these reasons, deep learning approaches are not considered in this work.

3. Methodology

3.1. Problem Specifications

The motivating application was the design of an automatic vending machine for bulk goods packaged into plastic bags to be used by a retailer. As noted in the introduction, large-scale industrial applications, warehouse management, and manufacturing scenarios are not of interest since there are already available solutions. The targeted use case also imposes the requirement for inexpensive computing architecture, with cost and computing-power limitations.

The problem addressed is depalletizing, which consists of removing goods from a pallet using a robotic arm. Of all the phases of the task, we focus on object identification to provide the location of the next best object to be removed. There is no assumption about the mechanical part (robotic arm and gripper). The robotic arm will be positioned according to the output of the algorithm, while the gripper will be tailored to the specific use case.

We assume the palletized items are roughly convex, but not of regular shape, although we expect them to be similar to each other (we refer to the previous Figure 1 for examples). Likewise, we assume that the items are organized into mostly horizontal layers, although not necessarily regular. This allows us to employ methods that do not rely on 3D vision, but simply require assessing the z-order of items. A rangefinder can then provide the actual z coordinate.

We also assume that the pallet is axis-aligned, which is a realistic assumption since it is a trivial task to axis-align a mostly rectangular target with image processing techniques. As for the items, when their geometric aspect ratio is not 1, we assume that they are mostly oriented along two directions. This is typically true for pallets containing items that do not have the same aspect ratio as the pallet itself and are organized so as to optimize their fit on the pallet's surface.

Since we do not target manufacturing scenarios, which may have to deal with complex shapes and possibly random orientations, we consider the exact orientation of the items to be irrelevant. This will make it sufficient to locate items by their bounding box.

3.2. The Object Detection Methods

Pattern Matching is a classic algorithm in image processing that involves comparing a predefined template against different regions of the target image. This technique is particularly effective when the appearance of the object is well defined and can be represented by a template. However, its performance may be influenced by factors such as variations in scale, rotation, illumination, and background.

The **scale-invariant feature transform** (SIFT) is a robust algorithm that operates by identifying distinctive local features, or key points, within an image that is invariant to changes in scale, rotation, and illumination [4]. The SIFT generates descriptors that encapsulate the local image information around each key point, enabling efficient matching between key points in different images.

The **Oriented FAST and Rotated BRIEF** (ORB) algorithm is an efficient method for feature detection in image processing [6]. ORB starts by identifying key points using the FAST algorithm, assigns an orientation to each key point for rotation invariance, and extracts binary descriptors using BRIEF to represent local intensity patterns. This algorithm enables efficient matching between key points in different images.

Finally, the **Haar cascade classifier** is a machine learning-based object detection method widely used in image processing [35]. This classifier is trained on positive and negative samples of a target object, creating a model that can efficiently identify instances of that object in new images. The Haar cascade model excels in detecting objects with specific structural patterns, making them particularly suitable for tasks like face detection. Each of these computer vision algorithms offers unique strengths and limitations, and their suitability for depalletizing system implementation depends on the already mentioned indices, specifically tuned to take into account the time constant of edge computing tasks.

3.3. Experimental Setup

The experimental setup for this research involved the utilization of a low computing power *edge* solution, i.e., one that does not rely on remote resources (a centralized server, cloud computing) to provide computing power. The device was equipped with a camera and operated in a controlled environment. The Raspberry PI 4 acted as the primary computing platform, responsible for running the computer vision algorithms and processing the captured images. The camera was mounted in a fixed position, vertically placed above the pallet and roughly centered, ensuring consistent image capture and analysis.

The Raspberry PI 4 serves also as the central controller of our setup, orchestrating the various components. Images are captured at a resolution of 1280 $\times$ 720, RGB 24 bpp, using the PI camera model HBV-1708 with autofocus and a 2592 $\times$ 1944 maximum resolution (see Figure 2). The software was developed in Python 3.8 using OpenCV and other libraries under the Raspberry Pi operating system.

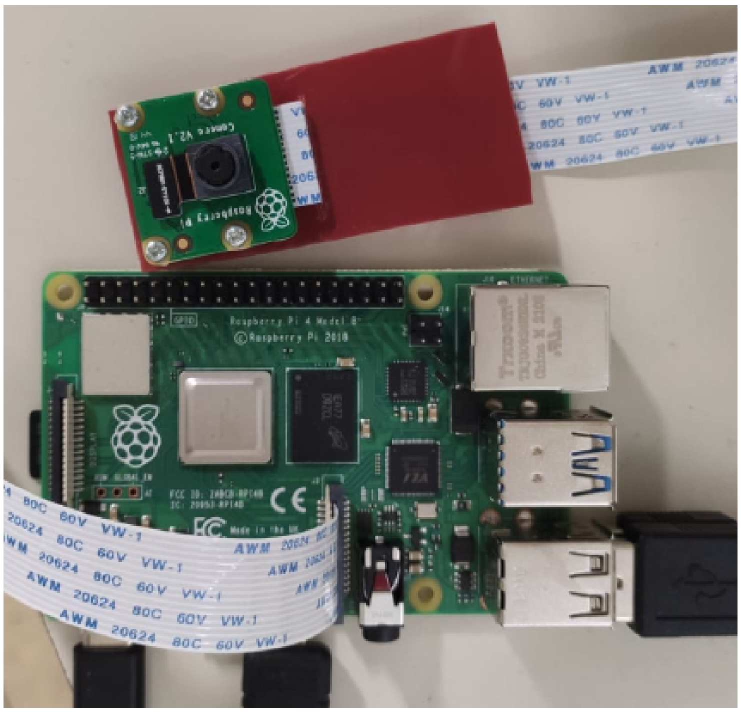
Figure 2. Raspberry PI 4 and camera.

Pallet items are detected on the Raspberry Pi 4, and non-maximum suppression [36] is applied to remove multiple overlapping observations. Then, the position of their center is estimated and in the final system can be transmitted to the robot to remove the identified item from the stack and place it onto the target position (which can be a conveyor belt or another pallet, depending on the task). Ideally, at each new iteration, the algorithm must guarantee that the topmost layer is emptied before moving to the next one. This can be achieved using a depth sensor (for instance, a rgb+d camera or a laser rangefinder). Since the setup did not involve a depth sensor, and since this function is implemented almost trivially by always targeting the object with the maximum z-order, the present work does not cover this aspect.

The experiments cover several feature extraction methods. The foreseen setup of the final equipment is illustrated in Figure 3.

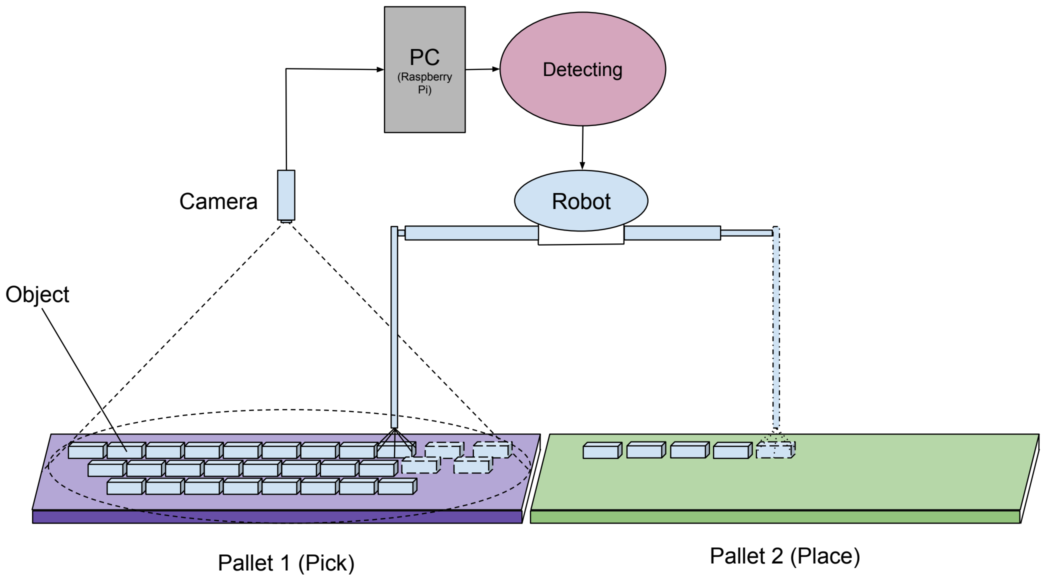

Figure 3. Reference setup.

In the rest of this paper, we will evaluate the performance of the four selected algorithms (pattern matching, Haar cascade, SIFT and ORB), comparing them based on their accuracy, speed, robustness to image variability, computer efficiency, detection sensitivity, and resource consumption. The methodology is outlined in Figure 4.

- Collecting the image data.
- Preprocessing and generating feature vectors using four feature descriptors, pattern matching, Haar cascade, SIFT, and ORB, individually.
- Comparing outcome algorithms.
- Selecting the most suitable algorithm for a specific depalletizing system.

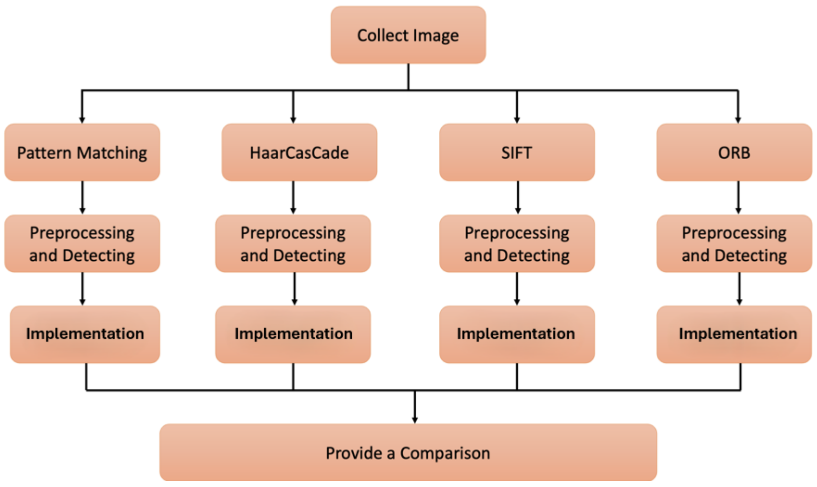

Figure 4. Experimental methodology.

3.4. Pattern Matching

Pattern matching is a common image-processing operation which detects and locates predetermined templates inside an image. With a special emphasis on the integration of template matching with non-maximum suppression, a widely used technique in this field, this Section provides some technical detail.

Pattern matching involves comparing a predefined template against different regions of the target image. In this research, the implementation of pattern matching followed these key steps:

1. **Template Generation**:Templates images are selected and augmented by applying rotations at various angles.
2. **Image Processing**: The system captured a grayscale image to simplify subsequent computations and facilitate robust feature extraction.
3. **Template Matching**: Image patches are compared to templates by using an appropriate similarity metric, discussed in the following.
4. **Non-maximum suppression**: Of a set of overlapping candidate regions, only the top scoring is retained.
5. **Result Visualization**

Image-template similarity is computed by normalized cross-correlation (NCC). This is the Pearson correlation coefficient [37], used in its 2D form and computed as follows:

$$R(x,y) = \frac{\sum_{i,j}(T(i,j)-\bar{T})(I(x+i,y+j)-\bar{I}_{x,y})}{\sqrt{\sum_{i,j}(T(i,j)-\bar{T})^2 \sum_{i,j}(I(x+i,y+j)-\bar{I}_{x,y})^2}}$$

where T is the template, I is the image, (x,y) is the origin of the image patch being compared, $\bar{T}$ and $\bar{I}_{x,y}$ denote mean pixel value in the template and in the image patch, respectively.

The matchTemplate function from OpenCV [30] was employed, and a threshold was applied to filter out false positives.

Non-maximum suppression (NMS) removes redundant bounding boxes, ensuring only the most relevant boxes are retained. Given a similarity score matrix R obtained after template matching, NMS retains the score at any pixel location that is a local maximum in its 3×3 neighborhood; otherwise, it suppresses it by setting it to 0.

As a final step, the end result is then displayed, indicating the detected objects' centers and their corresponding identification numbers (Figure 5).

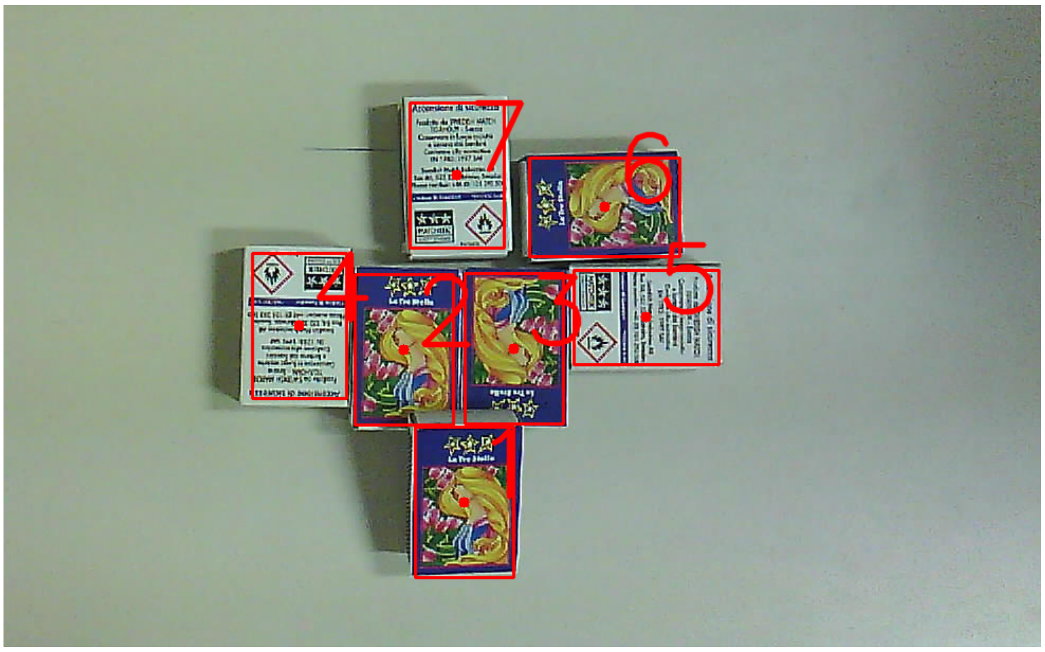

Figure 5. Pattern matching object detection technique tested on the matchboxes image.

3.5. Scale-Invariant Feature Transform (SIFT)

The scale-invariant feature transform (SIFT) is a robust algorithm for keypoint detection and feature matching in image processing [4]. This algorithm extracts the features of an object considering different scales, rotations, illumination, and geometric transformations. SIFT has been proven to be the most widely used algorithm in object recognition. It works in four phases as follows:

- Scale-space Extrema Detection;
- Keypoint Localization;
- Orientation Assignment;
- Keypoint Descriptor.

SIFT builds a multi-resolution pyramid over the input image and has proven to be very robust to noise and invariant to scaling, rotation, translation, and (to some extent) illumination changes.

In the code developed for this application, there are a few preprocessing steps applied to the images before performing object detection. The implementation of SIFT in this research followed these steps:

1. **SIFT Feature Extraction**: SIFT key points are detected and descriptors are extracted from key points in both the template and target images as histograms of local intensity gradients at multiple scales using differences of Gaussians.
2. **Looping Over Image Regions**: The code iterates over different regions.
3. **Matching and selection**: Matches between descriptors are selected, with a minimum match count.

4. **Filtering Matches**: A ratio test is applied to filter out good matches from the initial set of matches comparing the distances between the nearest descriptor and the second one in each descriptor.
5. **Homography Estimation**: a perspective transformation matrix M is estimated based the best-matched key points to align the template with the current region of interest.
6. **Perspective Transformation**: Transformation M is applied.
7. **Bounding Box Computation**
8. **Drawing and Visualization**: The bounding box and center of each detected object are visualized on the target image.

The result of detection is shown in Figure 6.

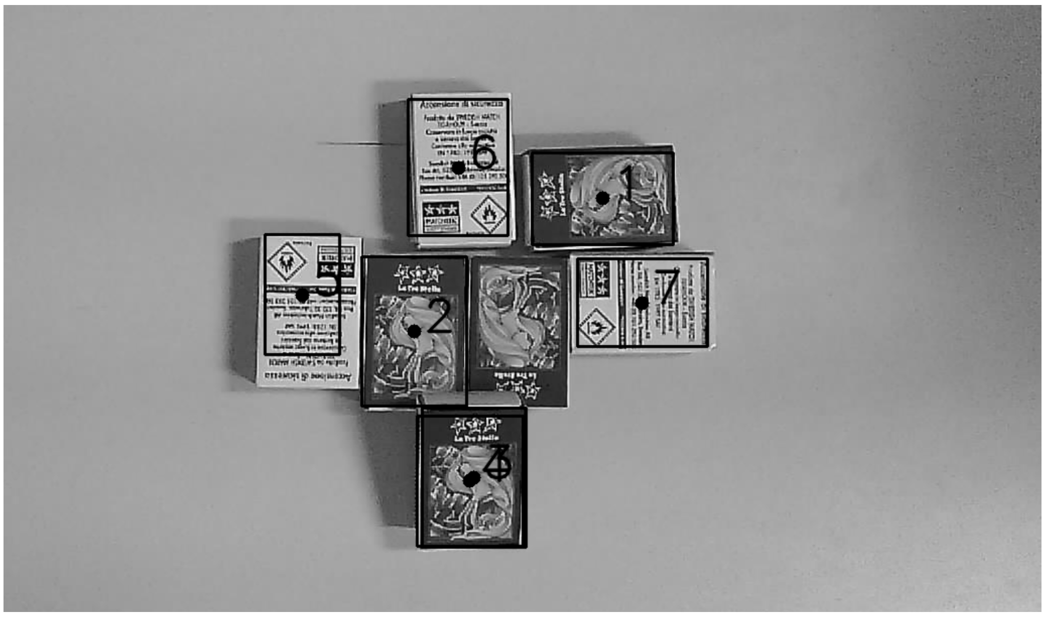

Figure 6. SIFT object detection technique tested on the matchboxes image.

3.6. Oriented FAST and Rotated BRIEF (ORB)

The Oriented FAST and Rotated BRIEF (ORB) algorithm is an efficient method for feature detection and matching in image processing, presented by Rublee and colleagues [6]. Compared to the SIFT and SURF, ORB is substantially faster in the usual situation.

ORB is a robust and efficient method for feature detection in image processing. It starts by identifying key points using the FAST algorithm, which efficiently locates areas with significant intensity changes. Unlike traditional FAST, ORB assigns an orientation to each key point, ensuring rotation invariance. The algorithm then extracts binary descriptors using BRIEF, representing the local intensity patterns around each key point. These descriptors enable efficient matching between key points in different images.

As in the SIFT, the matching process involves comparing the binary descriptors of key points in the target image with those in a reference template, then filtering out unreliable matches, and finally applying non-maximum suppression. The final result is a representation of detected objects with bounding boxes.

The ORB implementation in this research

1. implements the ORB detector for feature extraction in both the template and target image;
2. applies a sliding window approach with a defined step size for efficient detection;
3. employs a Brute-Force Matcher with Hamming distance for descriptor matching;
4. filters matches based on a predefined threshold;
5. performs non-maximum suppression to merge nearby bounding boxes;
6. marks the center of each bounding box and assigns a unique identifier.

The result of detection is shown in Figure 7.

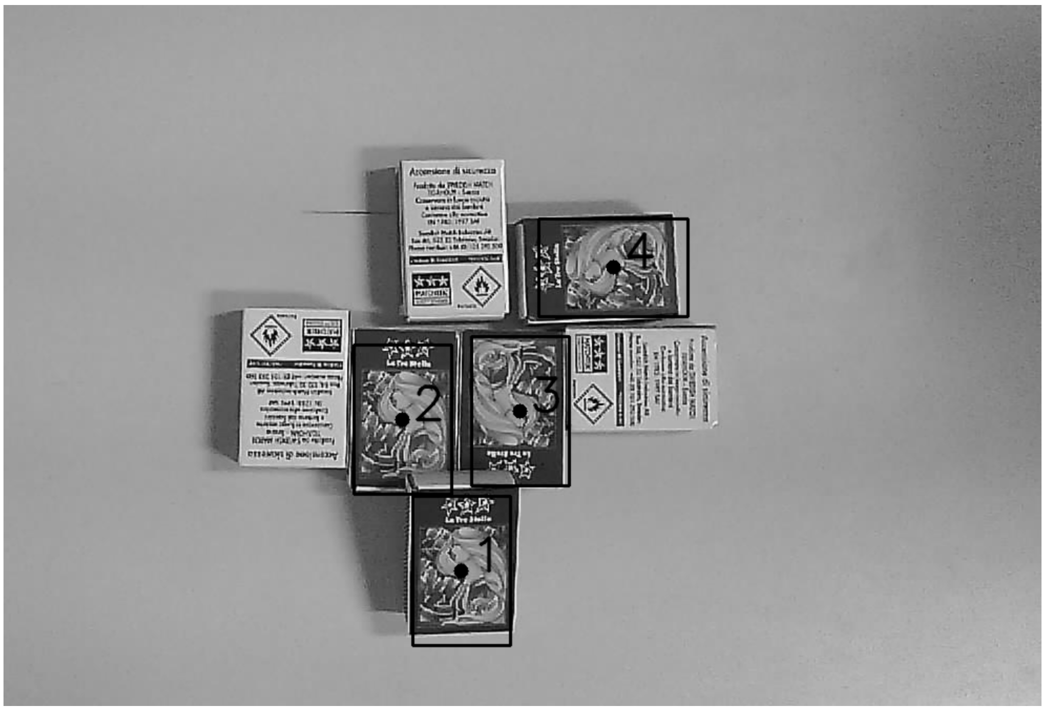

Figure 7. ORB object detection technique tested on the matchboxes image.

3.7. Haar Cascade Classifier

Haar cascade is a machine learning-based object detection method widely used in image processing. It is a classifier trained on positive and negative samples of a target object. Classification is based on features provided by rectangular filters.

Once trained, a Haar cascade classifier is applied to an image through a sliding window approach. Features are efficiently compared at different scales and positions. If a region of the image matches the learned pattern for the object, the classifier identifies it as a positive detection. The method is computationally efficient, but may have limitations in handling complex backgrounds or objects with varying orientations, as the training data need to encompass diverse instances of the target object for optimal performance.

The Python code used trains a Haar cascade classifier with OpenCV, creates a positive picture and various negative examples, and generates positive examples [35]. The primary steps are listed as follows:

1. **Prepare Negative Images**: A folder for negative images was created by copying images from a source directory. A background file (bg.txt) listed the negative images.
2. **Resize and Edit Images**: Positive and negative images were resized to consistent dimensions (the size 93 $\times$ 142 pixels was used for positive images).
3. **Create Positive Samples**: The *opencv_createsamples* tool generated positive samples and related information for each positive image in a separate directory for each image's data. The default configuration includes parameters like angles, number of samples, width, and height.
4. **Merge Vector Files**: Positive sample vector files were merged into a single file for training input.
5. **Train Cascade Classifier**: The *opencv_traincascade* tool trained the classifier.
6. **Completion**: After training, object detection is performed.

The resulting output is shown in Figure 8.

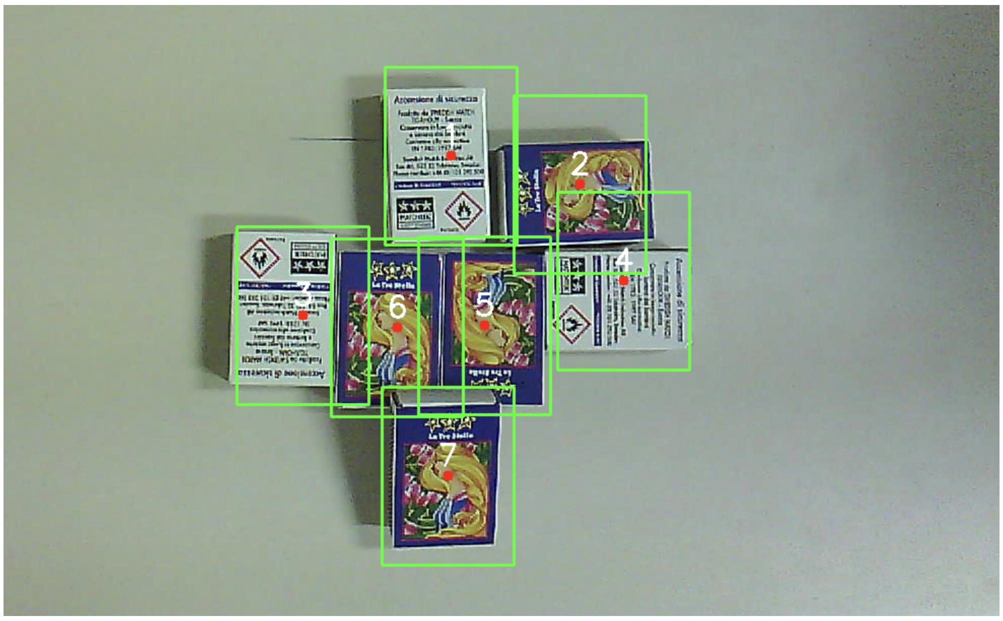

Figure 8. Haar classifier object detection technique tested on the matchboxes image.

4. Results and Discussion

In this section, we will present the performance evaluation of each computer vision technique based on several key factors as follows: accuracy, robustness to variability, computer speed, detection sensitivity, and resource consumption. We refer to Table 1 for quantitative results; this Section discusses them in the light of the problem requirements.

4.1. Accuracy

- Pattern Matching: This achieved high accuracy in object detection, with straightforward configuration by adjusting a single threshold and angle.
- SIFT: This demonstrated efficiency in finding key points, especially effective in rotation scenarios, contributing to its versatility across various applications.
- ORB: This maintained reliable detection accuracy for the front side of objects under certain conditions but showed limitations in recognizing the back part of matchboxes.
- Haar cascade: Despite a time-intensive training process, this exhibited only acceptable accuracy.

4.2. Robustness to Variability

- Pattern Matching: This demonstrated robustness to variability, showcasing resilience to changes in object appearance, lighting, and orientation.
- SIFT: This proved robust against scale, rotation, and illumination changes, contributing to its adaptability in diverse conditions.
- ORB: This displayed limitations in recognizing specific object orientations, impacting its robustness to variability. However, it remained reliable under certain conditions.
- Haar Cascade: This showed resilience to variations in object appearance and lighting conditions, contributing to its effectiveness in real-world scenarios.

4.3. Computing Speed

- Pattern Matching: This achieved fast detection speed, taking only a few seconds for implementation.
- SIFT: This boasted a fast implementation with efficient key point detection, contributing to its real-time applicability.
- ORB: This exhibited slower execution speed, contrary to expectations for a binary method, suggesting potential performance optimizations.

- Haar Cascade: This demonstrated quick detection post-training, with the inevitable and initial time investment required during the training phase.

4.4. Detection Sensitivity

- Pattern Matching: This exhibited sensitivity to changes in the detection threshold, offering flexibility in configuration.
- SIFT: This showed sensitivity to parameter adjustments, with a relatively quick tuning process.
- ORB: This displayed sensitivity to object orientation, requiring careful parameter tuning for optimal performance.
- Haar Cascade: This required attention to parameters such as setting variation and rotation angle, contributing to the time-consuming tuning process.

4.5. Resource Consumption

- Pattern Matching, SIFT, and ORB: These demonstrated efficient resource consumption, making them suitable for practical applications.
- Haar Cascade: This required significant computer resources during the training phase, with efficient resource consumption during detection.

The comparative analysis of the four algorithms—pattern matching, Haar cascade classifier, SIFT, and ORB—reveals distinct strengths and limitations across various performance indices and computational aspects. The results are summarized in Table 1.

Table 1. Performance indices of the implemented algorithms

	Training Time (h)	**Latency (s)**	**Total Matches**	**Precision**	**Recall**	**F1 Score**
Pattern Matching	–	0.13	7	1.00	1.00	1.00
Haar Classifier	3.55	0.09	7	1.00	1.00	1.00
SIFT	–	0.39	6	1.00	0.86	0.92
ORB	–	12.06	4	1.00	0.57	0.73

Training Time

- Pattern Matching, SIFT, and ORB: These algorithms do not require training, making them advantageous in scenarios where rapid deployment is needed.
- Haar Classifier: This requires a substantial training time of 3.55 h, indicating an initial setup cost. However, this investment pays off with excellent detection performance.

Detection Latency

- Haar Classifier: The fastest detection time (0.09 s) highlights its efficiency post-training.
- Pattern Matching: The quick detection time (0.13 s) without the need for training makes it a strong candidate for real-time applications.
- SIFT: The moderate detection time (0.39 s) reflects its computer complexity due to the detailed feature extraction process.
- ORB: Surprisingly, ORB takes the longest detection time (12.06 s), which is unexpected for a binary feature descriptor. This may be attributed to implementation details or the specific test conditions.

Total Matches

- Pattern Matching and Haar Classifier: Both achieve the highest number of matches (7), indicating high effectiveness in object detection.
- SIFT: Slightly lower total matches (6), reflecting its robustness but also its selective nature.
- ORB: The lowest total matches (4), highlighting potential limitations in detecting all relevant objects, especially in more complex scenarios.

Precision, Recall, and F1 Score

- Precision: All four algorithms exhibit perfect precision (1.00), indicating that when they do make detections, they are consistently accurate.
- Recall: pattern matching and the Haar classifier achieve perfect recall (1.00), showing their ability to detect all relevant objects. SIFT has a slightly lower recall (0.86), while ORB has the lowest (0.57), indicating it misses more objects.
- F1 Score: The F1 Score combines precision and recall into a single index. Pattern matching and the Haar classifier both achieve the highest possible F1 score (1.00). SIFT has a respectable F1 score (0.92), while ORB lags behind at 0.73.

4.6. Additional Remarks

As it can be observed from Figure 7, ORB, while reliable for the front side of objects, shows limitations in recognizing the back part of boxes. However, it maintains acceptable detection accuracy under certain conditions. It displays limitations in recognizing specific object orientations, impacting its robustness to variability. However, it remains reliable under certain conditions as well. ORB exhibits the slowest execution speed in implementation, contrary to expectations for a binary method, suggesting potential performance optimizations, requiring careful parameter tuning for optimal performance. This technique is able to work with restricted resources.

The Haar cascade exhibits moderately effective detection performance, resulting in acceptable accuracy. However, it shows resilience to variations in expressions and lighting conditions, contributing to its effectiveness in real-world scenarios. Concerning its computer speed, it demonstrates quick detection post-training, with inevitable and initial time investments required during the training phase. It requires attention to parameters such as setting variation, rotation angle, and different thresholds contributing to the time-consuming tuning process. In this approach, significant computer resources are required during the training phase, with efficient resource consumption during detection in a large number of negative images and a few positive images.

5. Conclusions

This article presented the implementation of four computer vision algorithms, namely pattern matching, scale-invariant feature transform (SIFT), Oriented FAST and Rotated BRIEF (ORB), and Haar cascade classifier [1–8], on a single hardware platform with low computing power, consisting of a Raspberry Pi 4, and evaluates their performance on the task of box recognition and localization to assess their suitability in a depalletizing system. Each technique has been described in detail and the respective implementations have been outlined. The experimental results were analyzed, evaluating the performance of the algorithms in terms of accuracy, robustness to variability, computer speed, sensitivity of detection, and resource consumption.

The results showed that pattern matching achieved high accuracy in object detection with a simple configuration process. SIFT demonstrated robustness to variations in scale, rotation, and illumination, contributing to its versatility in various applications. ORB showed limitations in detecting certain object orientations, but remained reliable under certain conditions. Despite a long training process, the Haar cascade classifier showed effective detection performance and acceptable accuracy.

The choice of the most appropriate technique depends on the specific requirements of the particular depalletizing system, considering factors such as accuracy, robustness, computer speed, and resource constraints. Each method has shown strengths and limitations, and the optimal choice may vary based on the unique characteristics of the application.

Future research in this area could significantly benefit from exploring the integration of multiple computer vision techniques. By leveraging their respective strengths and mitigating their limitations, we can create more robust and efficient systems. "Accuracy" and "diversity" are two relevant keywords in this context [38]. For example, an integration of the pattern matching, SIFT, and Haar classifier models studied in this paper could be promising, each having low latency, good detection capability, and being very different from each other. It is also worth noting that the detection latency of the combined model would be no more than 1 s, which is entirely compatible with the time constants of the application at hand.

Additionally, as already discussed in Section 2.3, investigating advanced deep learning-based object detection methods and their applicability to depalletizing systems could further enhance the system's performance and robustness even when not running on a Raspberry.

Overall, this research contributes to the advancement of automated object detection and localization in industrial environments, paving the way for more efficient and reliable depalletizing processes, ultimately enhancing productivity and operational excellence in factory settings.

Author Contributions: D.G.: conceptualization, supervision, methodology, writing, drafting of the article, review and editing, M.F. and A.V.R.: software implementation, experiments and data curation; F.M., S.R. and A.C.: conceptualization, visualization, supervision, review. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data available on request from the authors.

Conflicts of Interest: The authors declares no conflict of interest.

References

1. Brunelli, R. *Template Matching Techniques in Computer Vision: Theory and Practice*; John Wiley & Sons: Hoboken, NJ, USA, 2009.
2. Piccinini, P.; Prati, A.; Cucchiara, R. Real-time object detection and localization with SIFT-based clustering. *Image Vis. Comput.* **2012**, *30*, 573–587. [CrossRef]
3. Guennouni, S.; Ahaitouf, A.; Mansouri, A. A Comparative Study of Multiple Object Detection Using Haar-Like Feature Selection and Local Binary Patterns in Several Platforms. *Model. Simul. Eng.* **2015**, *2015*, 948960. [CrossRef]
4. Lowe, D.G. Distinctive image features from scale-invariant key points. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
5. Wang, J.; Jiang, S.; Song, W.; Yang, Y. A comparative study of small object detection algorithms. In Proceedings of the 2019 Chinese Control Conference (CCC), Guangzhou, China, 27–30 July 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 8507–8512.
6. Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G. ORB: An efficient alternative to SIFT or SURF. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 2564–2571.
7. Brown, M.; Szeliski, R.; Winder, S. Multi-image matching using multi-scale oriented patches. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005; IEEE: Piscataway, NJ, USA, 2005; Volume 1, pp. 510–517.
8. Muja, M.; Lowe, D.G. Fast approximate nearest neighbors with automatic algorithm configuration. In Proceedings of the VISAPP 2009, Lisboa, Portugal, 5-8 February, 2009. **2009**, *2*, 2.
9. Widiawan, B.; Kautsar, S.; Purnomo, F.; Etikasari, B. Implementation of Template Matching Method for Door Lock Security System Using Raspberry Pi. *VOLT J. Ilm. Pendidik. Tek. Elektro* **2017**, *2*, 143. [CrossRef]
10. Wang, W.; Li, W.; Zhang, Z. A Parallel PCA-SIFT Algorithm Based on Raspberry Pi 4B. In Proceedings of the 2023 7th International Conference on Electronic Information Technology and Computer Engineering, Xiamen, China 20–22 October 2023; pp. 913–920.
11. Bhatlawande, S.; Nahar, S.; Mundada, S.; Shilaskar, S.; Shaikh, M.D. Driver Assistance System for Detection of Marked Speed Breakers. In Proceedings of the 2024 2nd International Conference on Advancement in Computation & Computer Technologies (InCACCT), Gharuan, India, 2–3 May 2024; pp. 618–622. [CrossRef]
12. KAYMAK, C.; UCAR, A. Implementation of Object Detection and Recognition Algorithms on a Robotic Arm Platform Using Raspberry Pi. In Proceedings of the 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), Malatya, Turkey, 28–30 September 2018; pp. 1–8. [CrossRef]
13. Kumar, V.P.; Aravind, P.; Pooja, S.N.D.; Prathyush, S.; AngelDeborah, S.; Chandran, K.R.S. Driver Assistance System using Raspberry Pi and Haar Cascade Classifiers. In Proceedings of the 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 6–8 May 2021; pp. 1729–1735. [CrossRef]
14. Greco, D.; Masulli, F.; Rovetta, S.; Cabri, A.; Daffonchio, D. A cost-effective eye-tracker for early detection of mild cognitive impairment. In Proceedings of the 2022 IEEE 21st Mediterranean Electrotechnical Conference (MELECON), Palermo, Italy, 14–16 June 2022; pp. 1141–1146.
15. Mohamed, I.S.; Capitanelli, A.; Mastrogiovanni, F.; Rovetta, S.; Zaccaria, R. Detection, localisation and tracking of pallets using machine learning techniques and 2D range data. *Neural Comput. Appl.* **2020**, *32*, 8811–8828. [CrossRef]

16. Bay, H.; Tuytelaars, T.; Van Gool, L. Surf: Speeded up robust features. In Proceedings of the Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, 7–13 May 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 404–417.
17. Gue, K. Automated Order Picking. In *Warehousing in the Global Supply Chain: Advanced Models, Tools and Applications for Storage Systems*; Manzini, R., Ed.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 151–174.
18. Zhang, J.; Xie, J.; Zhang, D.; Li, Y. Development of Control System for a Prefabricated Board Transfer Palletizer Based on S7-1500 PLC. *Electronics* **2024**, *13*, 2147. [CrossRef]
19. Okura Flexible Automation Systems Pte Ltd. Okura Robot Palletizer Models A1600III and A700III Brochure. Available online: https://okura.com.sg/pdf/RobotPalletizer.pdf (accessed on 2 August 2024).
20. Asea Brown Boveri Ltd. ABB Robotic Depalletizer Brochure. Available online: https://search.abb.com/library/Download.aspx?DocumentID=9AKK108466A9114 (accessed on 2 August 2024).
21. Horn, B.K. Patrick Winston and the MIT AI Lab Copy Demo (1970). Available online: https://people.csail.mit.edu/bkph/phw_copy_demo.shtml (accessed on 2 August 2024).
22. Ikeuchi, K.; Horn, B.K. The Mechanical Manipulation of Randomly Oriented Parts. *Sci. Am.* **1984**, *251*, 100–111.
23. Holz, D.; Topalidou-Kyniazopoulou, A.; Stückler, J.; Behnke, S. Real-time object detection, localization and verification for fast robotic depalletizing. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 1459–1466.
24. Chiaravalli, D.; Palli, G.; Monica, R.; Aleotti, J.; Rizzini, D.L. Integration of a Multi-Camera Vision System and Admittance Control for Robotic Industrial Depalletizing. In Proceedings of the 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 8–11 September 2020; Volume 1, pp. 667–674. [CrossRef]
25. Arpenti, P.; Caccavale, R.; Paduano, G.; Andrea Fontanelli, G.; Lippiello, V.; Villani, L.; Siciliano, B. RGB-D Recognition and Localization of Cases for Robotic Depalletizing in Supermarkets. *IEEE Robot. Autom. Lett.* **2020**, *5*, 6233–6238. [CrossRef]
26. Aleotti, J.; Baldassarri, A.; Bonfè, M.; Carricato, M.; Chiaravalli, D.; Di Leva, R.; Fantuzzi, C.; Farsoni, S.; Innero, G.; Lodi Rizzini, D.; et al. Toward Future Automatic Warehouses: An Autonomous Depalletizing System Based on Mobile Manipulation and 3D Perception. *Appl. Sci.* **2021**, *11*, 5959. [CrossRef]
27. Prasse, C.; Skibinski, S.; Weichert, F.; Stenzel, J.; Müller, H.; ten Hompel, M. Concept of automated load detection for de-palletizing using depth images and RFID data. In Proceedings of the 2011 IEEE International Conference on Control System, Computing and Engineering, Penang, Malaysia, 25–27 November 2011; pp. 249–254. [CrossRef]
28. Vu, V.D.; Hoang, D.D.; Tan, P.X.; Nguyen, V.T.; Nguyen, T.U.; Hoang, N.A.; Phan, K.T.; Tran, D.T.; Vu, D.Q.; Ngo, P.Q.; et al. Occlusion-Robust Pallet Pose Estimation for Warehouse Automation. *IEEE Access* **2024**, *12*, 1927–1942. [CrossRef]
29. Li, Y.; Qi, H.; Dai, J.; Ji, X.; Wei, Y. Fully convolutional instance-aware semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2359–2367.
30. Bradski, G.; Kaehler, A. *Learning OpenCV: Computer Vision with the OpenCV library*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2008.
31. Wuest, T.; Weimer, D.; Irgens, C.; Thoben, K.D. Machine learning in manufacturing: advantages, challenges, and applications. *Prod. Manuf. Res.* **2016**, *4*, 23–45. [CrossRef]
32. Bansal, M.; Kumar, M.; Kumar, M. 2D object recognition: a comparative analysis of SIFT, SURF and ORB feature descriptors. *Multimed. Tools Appl.* **2021**, *80*, 18839–18857. [CrossRef]
33. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
34. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
35. Viola, P.; Jones, M. Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001, Kauai, HI, USA, 8–14 December 2001; Volume 1, pp. I-511–I-518.
36. Bodla, N.; Singh, B.; Chellappa, R.; Davis, L.S. Soft-NMS–improving object detection with one line of code. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 5561–5569.
37. Rodgers, J.L.; Nicewander, W.A. Thirteen Ways to Look at the Correlation Coefficient. *Am. Stat.* **1988**, *42*, 59–66. [CrossRef]
38. Kuncheva, L.I. *Combining Pattern Classifiers: Methods and Algorithms*; John Wiley & Sons: Hoboken, NJ, USA, 2014.