ICT GROUP

Computer Vision: cost-effective model training with Digital Twin technology

Abstract The following paper describes the process that led us to detect the state of our warehouse with the use of computer vision. Computer vision problems can be solved using traditional vision algorithms or machine learning (ML) vision models. Traditional vision algorithms are easy to understand and have predictable results, but may struggle to perform complex tasks, while ML vision models provide high accuracy and scalability but require large datasets to be trained on, and might produce unpredictable results. This paper presents a solution to getting the training images by virtually generating synthetic training images in the game engine Unity3D.

1. Problem Statement

In scenarios where traditional sensors are not capable of measuring a desired metric or where deploying numerous sensors is impractical, computer vision models present a viable alternative as they utilize visual data captured by a camera sensor. This approach overcomes the limitations of conventional sensors (pysical constraints, limited flexibility), providing a more effective method for data extraction.

Datasets composed of real images can be challenging to compose due to several factors: they are often time-intensive and costly to collect, may lack sufficient diversity to represent a wide range of scenarios, and can raise privacy concerns, especially when involving sensitive or personal data.

2. Computer Vision

2.1 Definition

Computer vision is a field of artificial intelligence that enables computers to interpret and understand visual information, such as images and videos. This technology allows us to recognize objects, people, or animals within a scene and extract useful information for processing them.

2.2 Computer Vision Techniques

Depending on the task, computer vision algorithms can perform more or less in-depth investigations of an image. Some of the most common techniques (Figure 1) include:

- **Classification:** consists of analyzing the content of an image and attributing it to a class;

- **Object Detection:** allows the identification of one or more instances within an image;
- **Instance Segmentation:** is the evolution of semantic segmentation, in addition to assigning a class for each object makes sure to discern the different objects from each other;
- **Semantic Segmentation:** a label is assigned to each pixel so that all objects within an image are identified;

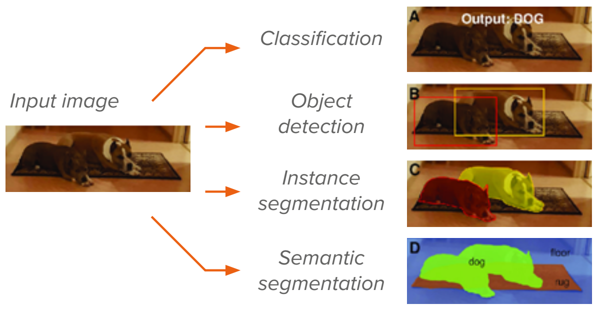


Figure 1: Computer vision techniques (Fazekas, Budai, Stollmayer, Kaposi, and Bérczi, 2022).

2.3 Computer Vision Algorithms

There are two main approaches to implementing computer vision to extract information: traditional computer vision algorithms and machine learning-based computer vision algorithms.

Traditional Computer Vision Algorithms

Traditional computer vision involves algorithms that contain a finite set of rules and operations in a programming language. The algorithm is then capable of performing only the operations for which it was designed and defined.

These computer vision algorithms methods rely on mathematical models and signal processing techniques to extract features from images or videos. One of the most widely used tools in traditional computer vision is OpenCV , also known as Open Source Computer Vision Library, which provides a set of functions for image processing, object detection, and feature extraction.

Compared with machine learning, this approach has advantages and disadvantages. Although it is easier to understand and demands fewer resources, it may require considerable manual effort to develop effective algorithms and be less flexible toward variations in visual data.

Machine Learning Computer Vision Algorithms

The Machine Learning technology is based on the concept that computers can learn from provided examples, identify the existing patterns and be able to make predictions autonomously, without having been explicitly programmed to do so.

Machine Learning systems originate from a dataset, a structured set of data organized into features and labels, which describe a certain scenario. The label is the quantity to be predicted, while the features are the properties on which the label's outcome depends.

The fundamental advantage of neural networks lies in their ability to improve performance as the amount of available data increases. The more data used to train the network, the better predictions it can make. However, to achieve optimal results, an effective model requires a large amount of data, which can range from hundreds to thousands of images, for example, in the case of visual recognition. Additionally, to process such large amounts of data efficiently and in a reasonable amount of time, powerful Graphics Processing Units (GPUs) with high amounts of memory are required.

3. The Problem

At the Center of Excellence, we have a warehouse within our factory (Figure 2), composed of 9 slots where workpieces in different colors can be stored. At the system's startup, we want to be able to determine what is stored in each warehouse slot automatically. This was not possible with the existing sensors already installed in the factory, and it was not possible to add additional sensors due to space constraints.

4. The Solution

Placing a camera pointing to the warehouse slots and training a computer vision model appeared to be the most effective way to detect changes in the warehouse. By doing this, we can reduce both costs and system complexity compared to adding numerous sensors.

Because the Center of Excellence developed a digital twin that virtually replicates the warehouse, we were able to generate virtual images (synthetic data) in the Unity3D game engine for training the computer vision model and use it to predict real-world images.

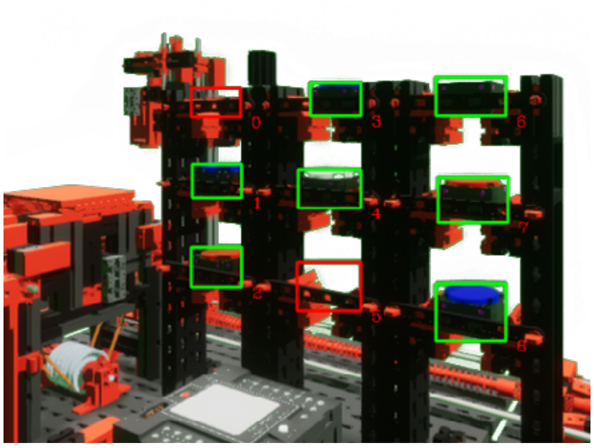
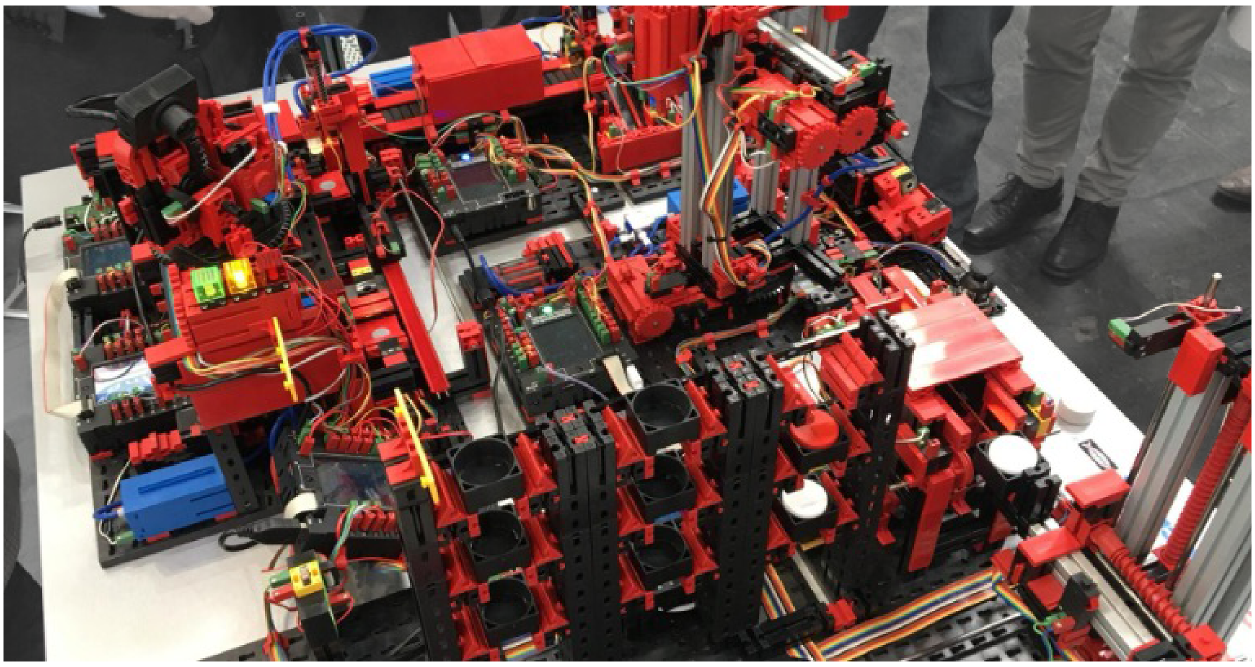


Figure 3: Detection of the warehouse slots

4.1 Dataset

To perform the detection effectively, computer vision systems need to be trained with as many properly labeled images as possible, which make up the training dataset. However, the larger the dataset, the longer the training time, so a compromise must be found between accuracy and time investment.

Figure 2: Picture of the factory.



Creating a sufficiently large dataset to train the algorithm (Section5.1) is in fact a very important aspect. It allows the training of a robust model able to make reliable predictions even in the presence of suboptimal conditions of light and resolution, deformation, partial coverage of the subject, or scale variations. To give an example, in the case of Image Classification, an ambiguous image might be associated with multiple classes, so the algorithm must try to minimize errors by assigning the label that guarantees the lowest probability of error.

4.2 Synthetic Data

By utilizing our digital twin to create synthetic data (Figure 4), we were able to generate tens of thousands of 1280 pixels x 1280 pixels images for the training dataset. All images were also automatically labeled based on object bounding boxes, significantly saving time and reducing the risk of human error in labeling. In real life, capturing tens of thousands of images using a camera and manually labeling each one would have required significantly more effort. This process would also have been prone to errors, such as incorrect or inconsistent labeling, which can be common in manual workflows. By automating this through synthetic data generation and automatic labeling, we drastically reduced the time required from hundreds of hours to just a few minutes while also ensuring greater consistency in the dataset.

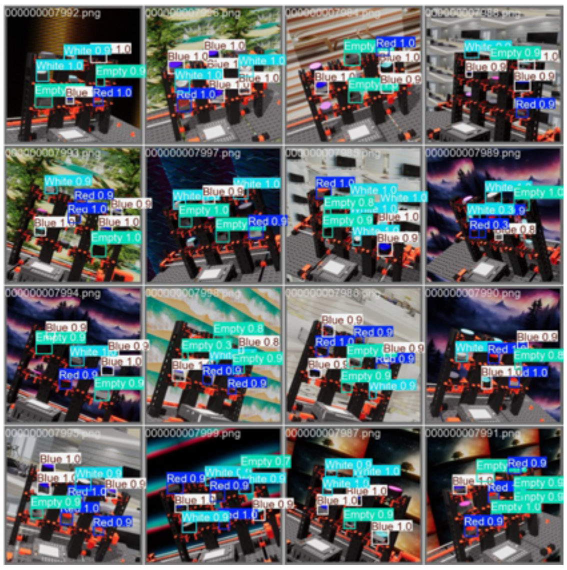


Figure 4: Synthetic data generated by the digital twin.

5. Methodology

5.1 Digital Twin

The game engine Unity 3D, was used in conjunction with the Unity Perception and HDRP packages to generate synthetic data with a realistic look.
The key to creating a robust and diverse dataset of synthetic data is to randomize as many parameters as possible. By varying parameters such as the virtual scene elements, lighting conditions, object properties, and camera settings, the synthetic data can more accurately mimic the complexities of real-world scenarios. This randomness ensures that the generated data covers a broad spectrum of possible conditions and interactions, which is crucial for training machine learning models that need to generalize well across different situations. Without sufficient variability, models may become overfitted to specific conditions present in the training data, leading to poor accuracy in real-world applications.

5.2 YOLOv8

YOLOv8 is a popular open-source Python library capable of object detection and segmentation in images.

The YOLO library reads the input image by dividing it into a grid of size N * N. In this way, the model is trained on the subsections of the images to be predicted rather than on the complete image. Thanks to this method, the recognition can also be done on images where only parts of the classified objects are visible.

As of now, there are ten versions of YOLO. Among these, YOLOv8 has demonstrated the best performance in our case in terms of accuracy.

YOLO also allows a technique called transfer learning, which means it is possible to train a model based on the existing knowledge of a previously trained model. Our approach utilizes the pre-trained YOLO variant called YOLOv8x-pose-p6, which is the largest in the series and supports images with a resolution of 1280px. Larger versions generally produce better results in most cases, but as they have more parameters, they require a bigger dataset for the training, more GPU memory and are slower to train and run.

5.3 Detection

To detect the state of warehouse slots, we used a Python script to capture images from a camera and automatically identify the state of each slot.
The script starts by drawing bounding boxes around nine predefined slots using manually defined coordinates. Next, the trained model is loaded into a Python script using YOLO. The script will then utilize the loaded model to perform real-time object detection, analyzing the input images or video frames.
The state of each slot is determined by checking if any detected object's bounding box overlaps with the predefined slot box. If there is an overlap, the slot is marked as occupied, otherwise, it is considered empty.

6. Results

As a result of the training, we were able to successfully train a model that could reliably determine the warehouse slots state, by only using synthetic data. The following paragraphs describe the results obtained during the process of solving the warehouse state problem.

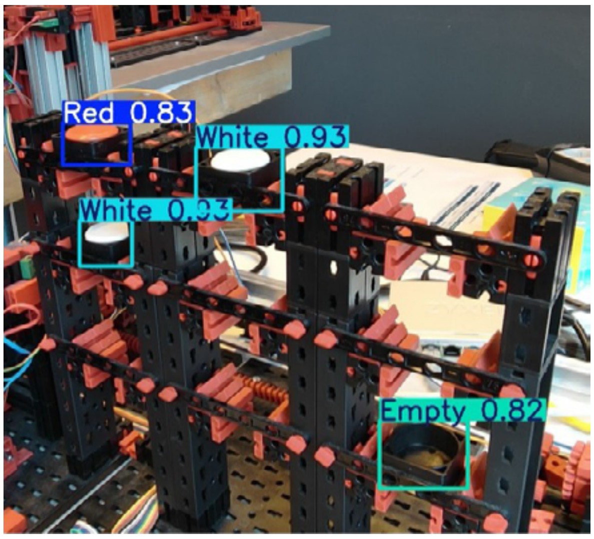


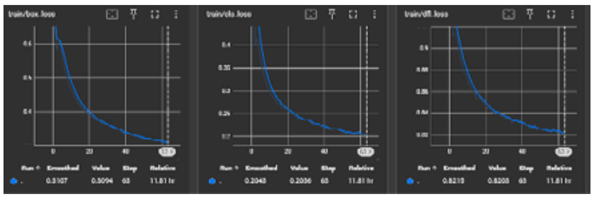
Figure 5: Warehouse slots detection with accuracy.

6.1 Hardware

For the training of machine learning models, GPUs are superior to CPUs because of their ability to handle parallel computations efficiently. Although our work laptop with 4GB of GPU memory could detect the workpiece using the trained model in a relatively short amount of time (an average of 0.12 seconds per image), it did not have enough memory to handle training. We have found two main providers that offer free cloud computing services with dedicated graphics processors through Jupyter Notebook:

- **Kaggle:** provides 30 hours of two Nvidia Tesla T4 GPUs.
- **Google Colab:** provides 1 Nvidia Tesla T4 GPU that is subject to quota restrictions and availability, meaning that the session could end at any moment while training.

Because of the reasons cited above, our decision was to use Kaggle as it could run the training for 12 hours, which was our desired amount of time (Figure 6). After this period, the model showed no signs of improvement.



Run	Smoothed	Value	Step	Relative
● .	0.3107	0.3094	63	11.81 hr
● .	0.2043	0.2036	63	11.81 hr
● .	0.8213	0.8208	63	11.81 hr

Figure 6: Accuracy improvement of a 12-hour training over time.

Challenges

Data Augmentation

Achieving the final result required several rounds of training, each presenting its own challenges. At times, certain colors failed to be detected, and in other cases, the accuracy dropped when the camera was rotated to particular angles. Overcoming these issues involved a process of trial and error after each training session. With each iteration, adjustments to the Unity scene (lighting, object colors, object positions, etc.) were made. This continued until the model's prediction accuracy was high enough during our tests. These results highlight the importance of tuning in the training process.

Black Box

A neural network often acts as a "black box", meaning that while it can provide accurate results, understanding how or why it arrived at a specific decision can be difficult. This lack of clarity can make it challenging to understand the outcomes of the computer vision models.

Camera Quality

One adjustment we needed to make to the hardware used for detecting warehouse slot states involved the camera choice. The images taken with the initial camera was not producing satisfying results, so we replaced it with a higher-resolution model that includes auto-focus and auto-exposure features, which led to better outcomes.

Conclusion

In conclusion, we were able to successfully train a computer vision model to detect the state of the warehouse by analyzing inputs captured with a camera, eliminating the need for extra sensors in our factory. The model was trained exclusively on synthetic data, saving us hundreds of hours that would have been spent manually capturing and labeling real images. Despite the model being trained on virtual images, it has demonstrated robust performance when tested with real-world inputs, working well under different lighting conditions and camera angles.

ICT High Tech
Center of Excellence
centerofexcellence@ict.nl
+31 (0)88 908 2000

ICT High Tech