An AI-based collaborative Robot System for Technical Education

Tobias Schubert¹, Sebastian Heßlinger², Alexander Dwarnicak³

Abstract

In this paper a cobot system is presented, that extends a Universal Robot with Artificial Intelligence (i.e., machine learning techniques) to allow for a safe human-robot collaboration, which is one of the main technologies in Industry 4.0 and is currently significantly changing the shop floor of manufacturing companies. Typically, these cobots are equipped with a camera to dynamically adapt to new situations and actions carried out by the worker who is collaborating with the robot in the same workspace. But obviously, switching from traditional industrial robots (acting completely isolated from humans) to smart robots also requires a change concerning the skills and knowledge workers must have to be able to control, manage, and interact with such cobot systems. Therefore, the main goal of this demonstrator is to develop a hard- and software environment, enabling a variety of different training scenarios to get trainees, employees, and students familiar with the main technical aspects of such human-robot interaction. Besides hardware and software related aspects, the paper will also briefly address the learning content, which is on the one hand, the basics of robotics and machine learning based image processing, and on the other hand, the interaction of the various components to form a functional overall system.

Keywords

Cobots; Machine Learning; Human-Machine Collaboration; Technical Education

1 Introduction

Robotics has been established in many companies for decades as an efficient way to automate recurring work steps. Recently, it can be observed that so-called cobots are gaining significantly in importance. Here, humans and robots work together in the same area without separation or protective spaces. On advantage is that monotonous or very strenuous tasks that quickly tire humans – such as overhead work – can be taken over by the cobot. Humans in this scenario complete this “team” by contributing their experience and knowledge, so that the overall production efficiency can be increased [1, 2].

With this growing importance of cobots, however, there is also a growing need to train students and employees accordingly. In the following we present an AI-based collaborative robot system, that fits well for this kind of technical education. Based on a Universal Robot, a 3D camera, a powerful PC, and machine learning algorithms, various application scenarios dealing with human-machine interaction are possible. One typical application can be summarized as follows: Objects lying in front of

¹ Festo Didactic SE, Rechbergstr. 3, 73770 Denkendorf, Germany, tobias.schubert@festo.com
² Festo Didactic SE, Rechbergstr. 3, 73770 Denkendorf, Germany, sebastian.hesslinger@festo.com
³ Festo Didactic SE, Rechbergstr. 3, 73770 Denkendorf, Germany, alexander.dwarnicak@festo.com
the cobot (either in inlays at fixed positions or randomly located somewhere on the desk) must be first identified and gripped by the cobot, before they are then handed over to the worker who – for example – manually inserts them into a larger product. Besides this kind of “stand-alone” mode, the demonstrator could be also integrated into an (existing) Festo Didactic learning factory. In this mode, the typical setup is based on the idea, that defective workpieces should be taken out of the production process by the cobot and handed over to a worker for a rework step. In both scenarios one major challenge is to localize and to follow the hand of the worker before placing the workpiece as accurate as possible into the worker’s hand, which is carried out by using machine learning based object detection algorithms in combination with a 3D camera.

One major challenge was to provide a single demonstrator for different levels of training, ranging from vocational schools on the one hand to employees already working in companies and universities / research projects on the other hand. Therefore, the entire system is developed as being “ready-to-run” when shipped, allowing trainees (without prior experience in the field of robotics and machine learning) to get to know how human-machine collaboration might look like. At this level, there is no need to re-program the cobot or the machine learning algorithms. Even switching between one kind of objects to be gripped to another set of objects can be carried out without programming a single line of code. But since both, the cobot programs and the machine learning algorithms are not offered as a black box but plain source code, students are encouraged to play around with different use cases, communication schemes between the cobot and the software tools, and to come up with completely new ideas.

The remainder of this paper is organized as follows: Section 2 gives a brief overview of the hardware components and software tools used in this project. Section 3 highlights some application setups that have been realized in the last months, followed by implementation details – especially regarding the so-called “hand-eye-calibration” – being discussed in Section 4. In Section 5, potential training scenarios are sketched, while Section 6 concludes the paper with a summary of the work done so far and some ideas for future work.

2 Hardware & Software

Figure 1 shows a schematic of the entire demonstrator. Concerning the hardware, the system mainly consists of the following components:

- Universal UR3/5e cobot [3]
- Vacuum, soft, or parallel gripper
- Intel Realsense D435 depth camera [4]
- Powerful PC with NVIDIA RTX 3060 graphics card

The cobot and the machine learning PC are connected to each other via an Ethernet cable, while the 3D camera is connected directly to the PC via USB. Regarding the software, especially for the implementation of the machine learning algorithms and the communication with the cobot, the following tools are used, among others:

- Python 3.9.0 & NVIDIA CUDA 11.3 development kit
- OpenCV, Torch, Flask, and some more libraries
- Robotics Toolbox / Swift 3D visualization [5]
- UR Real Time Data Exchange [6]
With the help of the aforementioned “Robotics Toolbox / Swift 3D visualization” it is possible – as Figure 2 shows – to highlight both the current orientation of the robot and the position of the human hand (represented by the dot marked in blue). Furthermore, the working area of the robot is visualized in transparent blue. For cobots that do not require a protective enclosure, the definition of a “safe” working area is of particular importance, as this defines the area in which no injuries can occur in the event of a collision with the worker. Conversely, the cobot is therefore not allowed to approach any position that is outside this safe working space. As sketched in Figure, the area just above the table, for example, is not part of the valid working space; the same applies to the immediate vicinity of the camera mount, in both cases to ensure especially that crushing of the human hand is impossible.

The valid workspace is defined within the UR cobot and is always queried before the cobot makes a movement. The main benefit of the visualization is to give the user graphic feedback as to whether, for example, his hand is in the safe working area and the cobot is therefore allowed to move to it or not.
On the machine learning PC, all routines have been realized with Python and the previously specified libraries. Especially the two libraries “UR Real Time Data Exchange” and “UR Extensible Markup Language Remote Procedure Call” play a major role, as they enable an efficient data exchange between the machine learning algorithms and the cobot – i.e., the current position of trained objects – in real time. On the UR side, the integrated graphical user & programming interface was essentially used to implement the corresponding functionality of the cobot. Both systems work in coordinated endless loops and typically run through the following steps:

1. “Wait for trigger signal” (Cobot)
2. “Localize object” (PC)
3. “Grab object” (Cobot)
4. “Localize and follow hand” (PC & Cobot)
5. “Place object into hand” (Cobot)

3 Use Cases

As already mentioned, the demonstrator can either be used “stand-alone” or integrated into an (existing) Festo Didactic learning factory. Figure 3 shows an example of the latter scenario, in which two UR3e cobots are used. The idea here is that the front cobot stretches out chassis of model cars to the user, the user then manually assembles the axles, and finally the chassis including the axles are brought into the further process implemented on the system. The rear cobot then mounts the vehicle bodies on the chassis. If an error occurs during assembly, the corresponding model car is removed from the process by the first cobot and is placed into the worker's hand to allow for some reworking.

In addition, numerous other use cases have been implemented in recent months, which usually require different grippers and are indicated in Figure 4: In addition to model cars, for example, different types of SD cards or rectangular workpieces can be gripped by vacuum grippers or parallel grippers and handed over to the worker, or even sweets/dice can be grabbed by a soft gripper and placed into a human hand.
In this section, some details regarding the algorithmic implementation are discussed. To do so, the following exemplary scenario is assumed: The cobot is located at a manual workstation of a Festo Didactic learning factory, where circuit boards arrive, into which an SD card of size 16, 32, or 64GB must be inserted. The cobot first waits for the trigger signal from the PLC (which is triggered as soon as a board is available and the required capacity of the SD card is known), then calls the machine learning PC to search for an SD card with the required memory capacity, picks it up and hands it over to the worker with the help of the hand detection routines running on the PC. The worker in turn inserts the card into the SD slot of the board and confirms the completion of the work step. Figure 5 illustrates this process (note, that the procedure sketched here is almost identical for other grippers, objects, and applications).

4.1 Hand-Eye-Calibration

In the scenario outlined above the algorithms on the machine learning PC are responsible for transmitting both the exact position of an SD card and the position of the human hand to the cobot so that it can follow the hand and eventually hand over the gripped object with high precision.

It is important to know that the coordinate systems of the camera and the cobot have different origins: The camera has its origin in the camera itself, while the cobot's coordinate system has its origin at its base. This means that all objects localized by the camera, or their corresponding x, y and z coordinates must be converted into the cobot's coordinate system, which is also known as “hand-eye-calibration” from literature. In principle, various implementations are possible, such as classic matrix transformations. But to be able to react flexibly to changing camera positions – for example, if the camera has shifted slightly due to transport – a more generic approach using machine learning-based regression was chosen.

For this purpose, a special, so-called ArUco marker is first attached to the robot (see Figure 6), which can be clearly and reliably identified by the camera with classic image processing routines. During calibration, the robot moves to several hundred positions (randomly) and briefly stops each time so that the camera can locate the marker. With each stop, the cobot not only knows its current position
in its own coordinate system, but the camera also localizes the marker with respect to its coordinate system. Furthermore, all application scenarios outlined here are chosen in such a way that the rotation of the robot in x, y, and z direction does not matter, i.e., it is assumed that the robot always grips or releases components vertically from above. This means that at the end of the positioning runs, several hundred pairs of \( <x, y, z> \) vectors are available in the cobot coordinate system, for each of which there is also a counterpart \( <x', y', z'> \) in the camera coordinate system. Finally, on the basis of this training data, an regression algorithm – in this case RANSAC [8] – is used, which ensures a sufficiently accurate mapping of the form \( M(<x', y', z'>) = <x, y, z> \) and is called by the object detection algorithms whenever the coordinates of an object or a human hand have to be transformed from the camera coordinate system into the coordinate system of the robot.

![Cobot with mounted ArUco marker for performing hand-eye-calibration.](image)

**Figure 6:** Cobot with mounted ArUco marker for performing hand-eye-calibration.

### 4.2 Machine Learning based Object Detection

As indicated before, during the typical workflow, object detection is responsible for locating specific objects on request of the cobot and for returning the current position of the human hand to the cobot. This task is performed by YoloV5 [9], a state-of-the-art method based on so-called deep learning with neural networks. To enable convenient operation, a web interface using Flask [10] is added, which provides the most important functions at the touch of a button (see Figure 7).

As usual for deep learning, training data must first be recorded and then annotated. In the scenario discussed here, this means for example saving images with SD cards and, in the second step, marking the SD cards shown on all images. In the third step, the neural network can then be trained and subsequently used for the actual application. The neural net used for this project has a total of 213 layers with about 7.1 million parameters. To simplify the change from one set of objects to another (e.g., from SD cards to sweets), two of the neural nets outlined above are used internally. One for the recognition of objects and one for the localization of human hands. This has the advantage that for
new applications, only the net for object detection needs to be (re)trained like sketched before, while the neural net responsible for hand detection can be retained as it is.

Figure 7: Object detection web interface.

5 Teaching Scenarios

The robot system presented here allows a variety of different experiments, from operating the demonstrator as such to reprogramming the initially given algorithms.

Regarding the cobot, first tasks are to get familiar with the user & programming interface provided by Universal Robots and to define, teach, and save various waypoints. Based on this, the next step is to specify the kind of movement (linear, constant, etc.) to be carried out when moving from one position to another. Furthermore, the trainees define the actions – such as opening/closing the gripper – that should be executed at each individual waypoint. Another important focus for collaborative robots should always be on safety aspects, i.e., it is important to think about reduced speed, safety stops, working space, collisions, and so on. Finally, the experience gained so far can be combined with control structures like variable assignments, loops, or if-then-else statements to come up with a cobot program in which, for example, a component is picked up at a certain position A and put down again at another position B.

There are also numerous training possibilities on the side of the machine learning applications running on the (separate) PC: A first task can be to localize new object types, i.e., to go through the entire workflow of “storing images”, “annotating images”, and “training & testing the neural network”. It is also conceivable to change the machine learning algorithms, to experiment with larger/smaller networks and to work out what the relationship is between the size of a neural network and its quality, or the runtime required to analyze a single image. Another interesting option is to replace the RANSAC algorithm used for “hand-eye-calibration” with other (regression) methods: Do these alternatives offer a more accurate coordinate system transformation?

Bringing both worlds together, the third series of exercises consists first of getting to know the data communication between the cobot and the machine learning algorithms with the help of the Python libraries provided by Universal Robots. This is of special importance to be able to forward position data of localized objects to the cobot at a high rate! Based on the existing framework, entirely new
applications are possible in a second step, such as sorting objects by color or stacking objects on top of each other according to a certain order. By the way, what about using machine learning to recognize dynamic gestures such as a “swipe to the right” and to move the cobot a few centimeters to the right (within its working space and safety constraints) according to the gesture shown by user?

6 Conclusion

In this paper we presented an AI-based collaborative robot system for technical education. Using a Universal Robot from the UR3/5e series, an Intel Realsense D435 3D camera, and a high-performance PC for the machine learning algorithms, a system has been realized that allows for getting familiar with aspects of human-robot interaction. As the object detection procedure, the state-of-the-art YoloV5 machine learning algorithm has been selected, which is able to recognize multiple objects within a few milliseconds, enabling real-time behavior. The entire demonstrator is easy to use and offers the possibility for numerous experiments, whether at vocational schools, in companies or as part of research projects. First feedback from customers and trainees are very positive.

As future work, adding another stereo camera directly at the cobot’s gripper is an interesting option. This would allow for even more precise navigation when getting close to an object. Having such kind of a camera system at the gripper would be also the “key enabler” for solving applications like the well-known “bin picking” task or when not only placing objects into a human hand but also grabbing arbitrary objects directly from the worker’s hand.

7 References