3.2

Construction of a Smart Vision-Guided Robot System for Manipulation in a Dynamic Environment

Janis Arents[1], **Modris Greitans**[1] **and Bernd Lesser**[2]

[1]EDI - Institute of Electronics and Computer Science, Latvia
[2]VIF - Virtual Vehicle Research GmbH, Austria

Abstract

This article outlines the construction of universal, user-friendly smart robot system for manipulation in a dynamic environment through AI-based vision system which incorporates processing on the edge. To successfully perform complex tasks in changing conditions, robots require both intelligence for adaptive decision-making and the ability to accurately perceive the environment and interface with it. The proposed system is built in a way that maximizes the modularity of the system. And thus, improves the ease at which the system can be modified to other specific goals after it has been operationalized. In this work, these characteristics are achieved by the use of synthetically generated data and Robot Operating System (ROS) as a middleware software. The first results prove the feasibility of training object detection networks on synthetically generated data sets. And also a combination of a 3D camera and industrial robot provides a convenient way for adding new objects to the database.

Keywords: edge computing, artificial intelligence, smart robot, smart manufacturing, synthetic data generation, robot operating system, computer vision, object detection, verification and validation.

3.2.1 Introduction and Background

Over the last decades, robots are increasingly used to off-load the physical labour of workers and to perform tasks more efficiently and accurately. This has resulted in a significant increase in productivity and quality of the performed tasks and manufactured products [1]. At first, robots were mainly used to take over the repetitive tasks of the workers. Today AI-based robotic systems are becoming an increasingly important part of manufacturing processes [2], [3], but industrial robots lack abilities to tackle the dynamic environment of today's manufacturing.

Manufacturing processes are equipped with a wide variety of sensors and cameras for quality control, safety fields, object detection in the 2D environment etc. Most of these processes are manually programmed for one specific task with only little tolerance for changes or adaption to different environments. Industrial robots in these systems are capable to manipulate with objects very precisely and repeat tasks with high accuracy. However, traditionally working in dynamic environments (especially with randomly distributed objects) still requires either human resources or dedicated sorting hardware. The latter one is usually spacious, expensive, and costly/time consuming to adjust if product assortment changes.

Changes in the marketplace translate into uncertainty for the manufacturing and end user mobility services. The way for business to succeed is by being flexible, smart, and effective in the manufacturing process [4]. However today many factories are still effectively designed for single purpose, that means there is little or no room for flexibility in terms of product design changes. In this article we propose a universal, user-friendly, and modular system that enables robots to "see" and work with randomly dropped objects that are overlapping with each other in a pile.

3.2.2 Challenges of Enabling Robots to "See"

The attention to picking and placing of arbitrarily placed objects that are overlapping each other in a pile has increased in the last years [4], especially in the context of Industry 4.0 and smart manufacturing [5]. The described problem is not only challenging to solve but also to be adapted and deployed to different manufacturing sectors.

3.2.2.1 Modularity

Different industries or manufacturing sectors have diverse conditions especially in the context of vision, such as lightning conditions,

environmental elements, and most importantly varied object types. The challenge within modularity includes efficient adaption to changes in the environment by respective component change such that a replacement of one component has no (or only minimal) impact on other components within the system.

3.2.2.2 Operability

Even though the modularity is a critical characteristic of a universal and flexible system, the operability plays an important role as well. Whereas the challenge with respect to operability includes user-friendliness and easy adaptability to other specified goals should be doable/manageable without any deep specific knowledge of the underlying target technology.

3.2.2.3 Computer Vision Algorithms

Two computer vision problems - object detection and instance segmentation - are sufficient to automate many tasks of an industrial robot. The detection indicates where in the camera's frame an object is located, and which class does this object belong to. Whereas segmentation determines which class does every pixel of an image belongs to. Instance segmentation is a type of segmentation that differentiates among pixels belonging to different instances of the same class. With this information, one can acquire a visible shape of a specific object and use it to determine an object's pose, which in turn is handy for picking up and manipulating the object. However, detection and segmentation are challenging tasks in the case of randomly piled objects. The objects are often only partially visible, and when the pile consists of similar or even the same object types, the similar features that could be used to detect the unobstructed objects are scattered all over the pile [6]. Similarly, an instance segmentation algorithm might struggle to distinguish similar and partially overlapping objects. Thus, the AI-based methods depend on annotated training data, where each new object requires numerous new training examples of pile images and labelling of such data is a tedious and very time-consuming manual labour, especially in the case of image segmentation tasks.

3.2.2.4 Validation of Algorithms

Since the advent of deep AI algorithms not only the performance but also the complexity of the respective algorithms has drastically increased.

The increased complexity of these algorithms in turn poses a unique set of challenges to both system designers and system validators. Since laboratory-based testing and user validation often forms a very time-costly and expensive task, simulation-based testing and user validation are often a preferred method in this aspect to (a) shorten development cycle times and (b) to reach a higher level of system maturity before testing and validating the system under laboratory- and real-world conditions.

Simulation- as well as laboratory testing and validation methods in general face both, a significant state space explosion problem as well as a gap to the real-world environment. For vision-based systems, this may arise from many aspects, for instance light conditions or dirty or distorted lenses or sensors etc. It is therefore crucial to system testers and validators to design their experiments not only as close as possible to the real-world conditions, but they must also be aware about the coverage of representative corner conditions and border cases which might affect the system in the field. Only in this way experiments can be designed to address many issues as possible beforehand, and to report valuable feedback to the systems designers during development cycles.

3.2.3 Requirements

To successfully perform complex tasks in changing conditions, robots require both intelligence for adaptive decision-making and the ability to accurately perceive the environment and interface with it. Enabling robots to "see" in terms of ability to work with objects that are different and unstructured in piles where industrial robot movements cannot be pre-programmed can support many workers in challenging working environments. However, this ability requires specifically designed solutions which addresses the stated challenges.

These needs introduce a sufficient degree of modularity to the system as a strong requirement to keep a high operability in both changing environments but also when changing system components: goal is to keep both the effort as well as cost as low as possible when adapting the system to e.g. other types of objects, changed environmental lighting conditions, but also when adapting the system to use other hardware components like other types of sensors, (which might feature different characteristics in terms of resolution, accuracy and even sensor failures like lens distortions, which might have to be handled differently for different sensors). To alleviate the training data acquisition

process and simplify the use of computer vision methods in industry the solutions require more efficient data preparation.

The extent to which a system can be decomposed into independent interacting modules that can be separately understood is beneficial not only with respect to Verification and Validation (V&V) efforts. Thus, to reduce complexity, independent and interchangeable system component-modules have to be defined that can be separately implemented, tested and validated to achieve a specific functionality. I.e. when the robot should be retrained to handle different kinds of objects under different environmental conditions, it should not be required to redesign or revalidate the perception system itself; when the perception system is being exchanged for another one, it should not be required to redesign and revalidate the module handling the user interaction or the module performing high level decision making. However, in such a case it is still important to validate the proper integration of the new module to provide assurance about the system as a whole.

The design of representative experiments featuring a high coverage of potential issues arising in the field requires a comprehensive standardization of the experiments with simultaneous preservation of degrees of freedom for adapting the experiments to modified use case requirements as well as to similar application domains. Thus, with respect to the proposed validation framework, we aim at accompanying the AI algorithm design- and training phase by providing a toolbox that allows for efficiently creating standardized representative experiments while being easy to handle and by being as intuitive as possible to the user.

Being based on existing, publicly available open source software building blocks, the validation framework should form a software abstraction layer easing the handling of a synthetic image generator to setup and conduct a variety of simulation scenarios (including physics simulation), the rendering of realistic image scenes as well as the generation of required ground truth annotation information (i.e. segmentation- and depth images/information) without having to directly deal with the complexity of the different underlying lower level software modules.

Furthermore, the toolbox shall come with a set of helper methods trying to ease (a) the generation of synthetic experiments (i.e. generated benchmark datasets), (b) the evaluation of the system-under-test's performance on generated datasets, while supporting the usage of hardware accelerators (i.e. GPUs) and to allow for parallel data processing using remote- and distributed computing. In addition, by using generic interface types, the framework shall be extendable with little effort to also incorporate other software modules

later on, like, for instance style transform networks to further decrease the simulation-reality gap or to also allow for applying adversarial dataset generation techniques.

3.2.4 Proposed Solution

The architecture of the proposed solution, Figure 3.2.1, is built in such a way that maximizes the modularity of the system and improves the ease at which the system can be modified to other specific goals after it has been operationalized. The experimental system is intended to be fully functional by the use of at least two 3D cameras and respective edge devices in a combination with an industrial robot. Moreover, the designed architecture does not preclude adding an additional camera-edge-robot blocks to supplement a pick and place process in scenarios when different objects are mixed in one pile and requires different grasping strategies. In following the sections hardware and software components will be described in more detail.

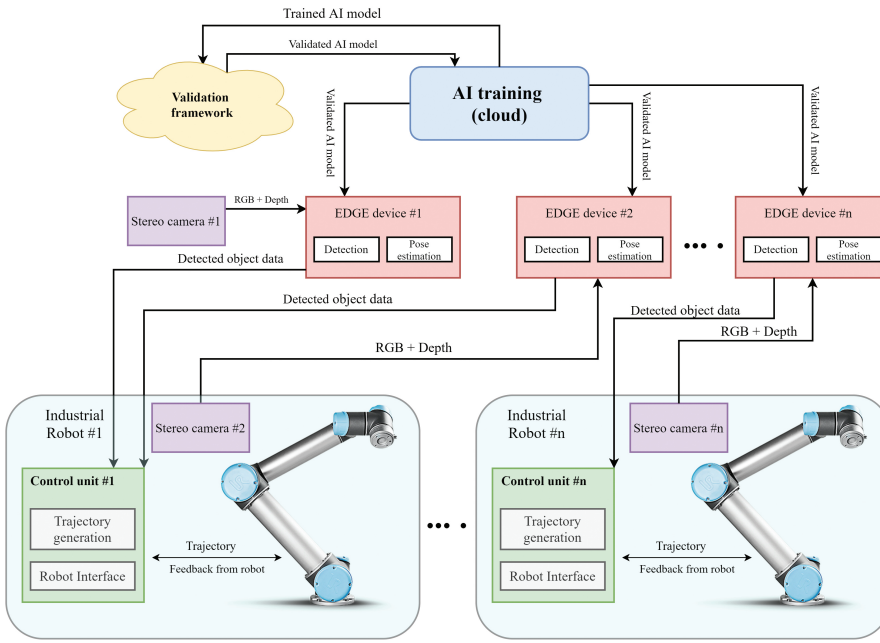

Figure 3.2.1 Architecture of the proposed solution.

3.2.4.1 Hardware and Interface Components

3.2.4.1.1 Robot Interface

Several interfaces are available to communicate with a robot [7]. The robot operating system (ROS) [8] is a popular abstraction layer to interface with a robot, and we propose that ROS is used for our proof-of-concept demonstrator. Using ROS for interfacing allows easier re-use for other robot types. A limitation of ROS is that no hard real-time constraints are supported. This will be addressed in ROS-2, a major rewrite of the ROS code. For a proof-of-concept, both ROS and ROS-2 are valid options, and over time it is expected that ROS-2 will become the preferred option for commercial products. ROS is supported for different operating systems – with Ubuntu Linux the main supported operating system.

3.2.4.1.2 Industrial Robot

Using ROS as an abstraction layer, a wide range of devices can be supported. For our proof-of-concept demonstrator, a Universal Robots UR5, 6 DOF (degreed-of-freedom) robotic arm is considered as a hardware platform for a smart robot. The maximum payload of UR5 can reach up to 5kg and the default reach is 850mm. Thus, the reach of the robot can be improved by gripper modifications. The repeatability of the UR5 robot is +/- 0.1mm.

3.2.4.1.3 3D Cameras

To enable robots to "see" we propose the usage of at least 2 3D cameras. One camera is statically mounted above the robot, perceives the environment around it and locates the region of the object of interest. The second camera is mounted on the robot as gripper modification for closer and more precise data acquisition from the object of interest. In this work, we have evaluated two different cameras for this task: a *Zivid One M* stereo camera [9] and an *Intel RealSense D415* stereo camera [10]. The *Zivid One M* stereo camera uses structured light as 3D technology, features a resolution of 1920x1200 and a common point precision of 60 μm and operates at up to 12 frames per second. The main parameters of the *D415* stereo camera are a resolution of 1280x720 for depth images and a frame rate of up to 90fps.

3.2.4.1.4 Deep Edge Device

The data acquired from the 3D cameras is then processed on deep edge devices. The evaluation board *ZCU102* [11] was utilized for prototyping and verification purposes and to define the necessary parameters and interfaces

for the carrier board. The carrier board is necessary for the "brains" of the edge processing unit (image and ML algorithm – System-on-a-Module). In essence, SoM is a bare-minimum board with all the necessary peripherals (e.g., RAM, power supplies, etc.) forming a stand-alone system. The SoM is connected to a carrier board, which has the necessary peripheral devices and connections for the use-case. By utilizing a SoM, it is possible to reuse the "brains" of the operation in different systems or to easily upgrade them in case there is an increased demand for performance. Therefore, the computing unit and its periphery is separated and can be upgraded independently. In our use case, the aim is to perform image and ML algorithm processing at the edge, therefore the developed carrier boards dimensions must be minimized. To interface with the stereo camera and other USB peripherals, an USB3 hub has been integrated into the carrier board. To forward the processed data to the robot control unit, the carrier board has gigabit ethernet connectivity to ensure a low latency connection with the control unit.

3.2.4.2 Software Components

3.2.4.2.1 Computer Vision Algorithms

The perception software module consists of an image and depth-map processing AI that segments images, detects pickable objects, and determines the orientation of the detected object. The detection is accomplished by a YOLO deep neural network architecture [12] and the segmentation is done by a Mask R CNN instance segmentation model [13]. To detect objects using YOLO the data from the camera must be pre-processed. The main step of pre-processing is object scaling regarding to the trained model, so that the object proportions are the same. YOLO detects all the objects in one frame on which the model has been trained on and then selects the best pickable object by the highest confidence rating. Additionally, the object's pick position is determined in the same frame and a name/ID is defined for the object. Thus, not all detected objects can potentially be picked by the robot arm, as the object could be too close to the side of the container, or the approach angle is too high. Therefore, different parameters are applied, and initial collision checking is done to decrease the *pick&place* cycle time and increase the picking success rate.

3.2.4.2.2 Synthetic Data Generation

Data preparation for machine-learning tasks plays an important role and according to Cognilytica [14] on average more than 80% of time spent on AI

projects are based on the collection and the processing of the data. The data collection techniques can be distinguished in several methods. The reviewed techniques published in [15] varies for different use cases. For example, in applications such as every-day object detection or machine translation there are publicly available data sets that could be reused and adapted for one's needs for model training [16]. In the context of smart factories, the situation is different, where product variety is changing more quickly, and algorithms must be repeatedly trained on new data sets. In these cases, re-usability of existing data sets is fairly low and manual labeling methods cannot meet the requirements of agile production as it is time-consuming, expensive and usually requires expert knowledge of the specific field. The most promising technique in terms of flexibility and comparatively low cost is synthetic data generation where time consumption is reduced depending on processing power and how optimized the generation algorithms are implemented.

Our proposed data generator itself consists of a python library encapsulating the 3D render engine specific commands for setting up and rendering the synthetic images and forms an easy-to-use abstraction layer supporting an application-specific set of image generation parameters. Since a vast number of synthetic images will be generated, the data generator library further provides hardware accelerator support (i.e. GPUs) wherever applicable as well as support for remote deployment and execution to ease massively parallel remote data generation. The current development version is implemented using Python 3.7 [17] and interfaces the open-source 3D render engine Blender® [18] v2.92.

3.2.4.2.3 Object 3D Reconstruction

The synthetic data generation framework is intended to be used with object 3D models, whereas in some cases the 3D models of the objects of interest are not available. For such situations object scanning software tools using depth cameras are being developed. The scanning software uses a camera mounted on the robot arm, capturing data from different viewpoints, to gather data from all sides of the object. The point clouds gathered from different viewpoints are aligned using the camera position information from the robot system. The alignment is fine-tuned by estimating a transformation for point to plane distance. After alignment, a 3D mesh is generated representing the object.

3.2.4.2.4 Validation Framework

The purpose of the validation framework is to automate the standardized procedure of performance evaluation (i.e., systematically carry out a vast

number of deterministic test inferences using the AI algorithm under test) on the generated datasets and to perform corresponding bookkeeping about the achieved object detection results. The results further serve as valuable information for the designers of the AI-based object detection algorithm during repeatedly improved training- and testing iteration cycles.

The validation framework is being implemented in Python 3.7 featuring a distributed master-worker architecture. It interfaces the synthetic data generator and the AI algorithm under test, a bookkeeping module for storing the validation results and an easily expandable plug-in system for applying specific analytics to the device under test.

3.2.4.2.5 Robot Control

After the object's pick position and orientation is determined a collision-free trajectory is generated, including different pose generation for approaching any detected object and for successfully picking it. A time optimal trajectory generation is used for the generation of trajectories with smooth and continuous velocity profiles. Moreover, several common ROS packages are used for ensuring the modularity with different sensor types and robots. ROS-Industrial is used to extend the advanced capabilities of ROS software to industrial relevant hardware and applications. For example, for interfacing with Universal Robot UR5 driver enabling ROS operation of UR robots is used. Moreover, the MoveIt! [19] motion planning framework that runs on top of ROS is utilized for robot arm navigation, motion and trajectory planning, robot interaction etc.

3.2.4.3 Hardware/Software Partitioning

The main functions of the proposed solution - object detection and pose estimation - will be performed on the deep edge device. Moreover, the pre-processing will be done using an application processing unit APU, but neural network models will be deployed on DPU. Furthermore, the object pose, and type will be sent to the control unit for trajectory generation to pick up the detected object. An additional application server will be used for application interface/GUI, training and validation supervision, edge configurations and implementation of the trained AI model and other applications. The object 3D reconstruction utilizes an industrial robot in a combination with stereo camera to precisely acquire data of the object of interest. The detailed HW/SW partitioning can be seen in Figure 3.2.2.

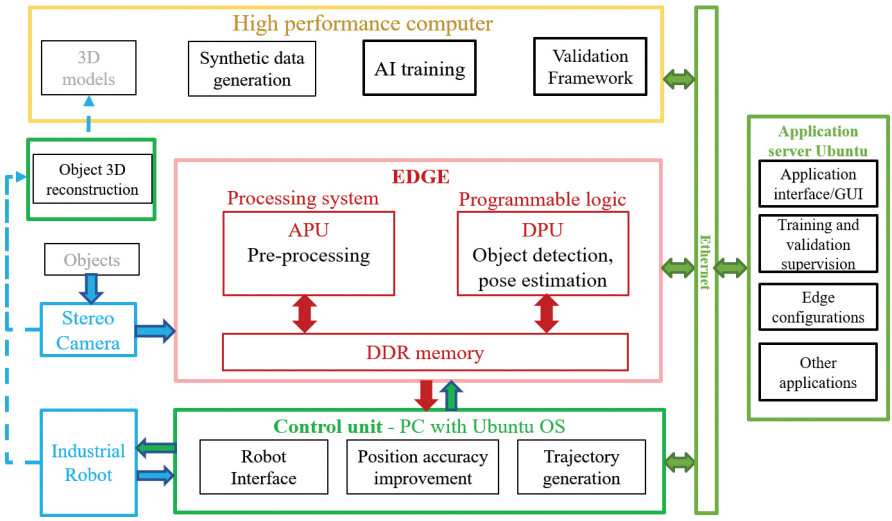

Figure 3.2.2 Hardware/Software partitioning.

3.2.5 Demonstrator Setup and Initial Results

Firstly, the following dataset is generated: for every individual (independent) scene, we fill an initially empty box with 50 randomly placed objects by making use of Blender's physics simulation engine to achieve realistic positioning and orientation. We use textures and Blender's principled BSDF shader nodes to achieve realistic renderings of the scenes including reflections. After filling the box with the objects, we create a series of (data dependent) renderings both varying 4 different light power levels and the orientation of the camera, which orbits around the box and renders the scene from 16 different angles, which can be seen in Figure 3.2.3. For every camera orientation, we also generate a depth image and the segmentation images of the individual objects (Figure 3.2.6) as seen by the camera and labelled by the object ID. We further generate an annotation file for every camera perspective which contains the individual object's orientation and rotation in camera coordinates together with the object's visibility percentage.

First results of a successful implementation on EDGE device of the trained YOLO model on synthetically generated data (Table 3.2.1) is achieved by using only one object type, where the model has been trained to detect only fully visible objects that are not obstructed by other objects. The test data consists of real 300 images.

Figure 3.2.3 Renderings with different light power levels and camera orientations.

Table 3.2.1 First object detection results, where the model has been trained on syntheticaly generated data.

mAP@0.90	mAP@0.75	mAP@0.50	Synthetic data set		Best result
			Total	Augmented	(steps)
44.99 %	94.35 %	96.73 %	4000	16000	6400

Accordingly, the whole workflow of the system has been tested, as it can be seen in Figure 3.2.4, where the AI is used to analyse the data of the Intel RealSense camera, which incorporates processing on the edge (FPGA based SoC). Currently the object detection is done on the edge device and then the further processing such as: object segmentation, pose estimation and communication with the robot is done separately on another processing unit (application server).

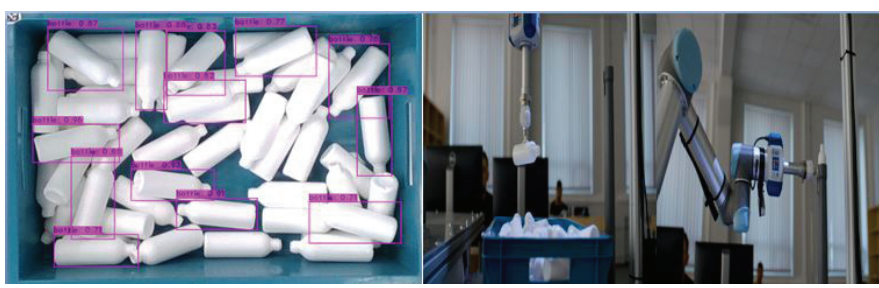

Figure 3.2.4 Object detection and pick and place operation.

Figure 3.2.5 Setup for object 3D reconstruction.

Furthermore, the first results on adding new objects to the scene have been also achieved. The Zivid 3D camera mounted on the robot arm is utilized to reconstruct the 3D model of the object of interest (marked in red in Figure 3.2.5). Depending on the dimensions of the object, the robot moves in a certain distance and angle around the object to precisely acquire a point cloud and generate a 3D model.

The reconstructed 3D model is then added to the synthetic data generation software by mixing different kind of object types in a pile. The next version of the dataset will use 2 different 3D models of objects for filling the box, where half of the objects are matt white plastic bottles, and the other half are shiny aluminium metal cans Figure 3.2.6.

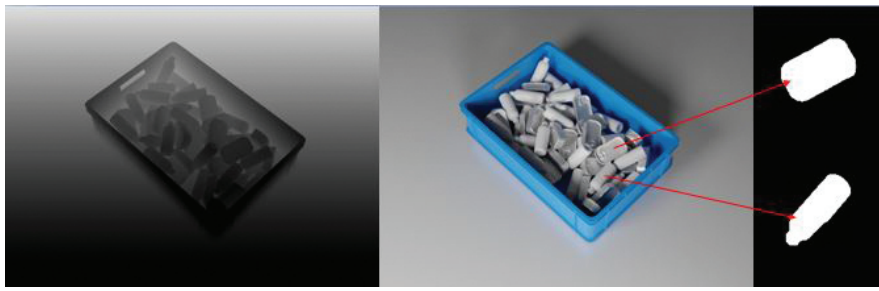

Figure 3.2.6 Scene including the plastic bottle- and the reconstructed metal can 3D models (middle) together with the and corresponding depth image (left) and segmentation masks (right).

3.2.6 Conclusion and Future Work

The proposed system consists of several elements that tackles the challenges of enabling robots to "see". The first results prove the feasibility of the proposed system and form the basis for further developments within the AI4DI project. The proposed hardware and software components leverage the modularity, operability, and the functional correctness of the system. Even though the results of using only synthetic data for training AI-based computer vision algorithms are promising, different combinations of synthetic and real training data sets will be explored as well. Furthermore, future work will include improvements of the object detection algorithms and continued improvements on the used pose-estimation methods. In scenarios where various objects are mixed in one pile different grasping strategies could be required, a case for which multi-robot collaborations methods are being explored and will be implemented during the project.

With respect to the synthetical data generation framework, in this project we are developing a set of open-source software building blocks to automatically generate a large amount of photorealistic training- and validation data for our robotic bin picking use case. The dataset images are fully annotated with position- and rotation information, including depth images, a labelled segmentation mask as well as a visibility score for every object visible in the scene. We believe that our set of software building blocks can be easily adapted or extended and allow for a rapid creation of similar datasets also for other industrial applications. The datasets created during this project will finally be made publicly available on the project website.

Acknowledgements

This work is conducted under the framework of the ECSEL AI4DI "Artificial Intelligence for Digitising Industry" project. The project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826060. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Austria, Czech Republic, Italy, Latvia, Belgium, Lithuania, France, Greece, Finland, Norway.

References

[1] Lee, K. "Artificial intelligence, automation, and the economy." Executive Office of the President of the USA 20 (2016).

[2] Carbonero, Francesco, Ekkehard Ernst, and Enzo Weber. "Robots worldwide: The impact of automation on employment and trade." (2020).

[3] Evjemo, Linn D., et al. "Trends in Smart Manufacturing: Role of Humans and Industrial Robots in Smart Factories." Current Robotics Reports 1.2 (2020): 35-41.

[4] Alonso M, Izaguirre A, Graña M. Current research trends in robot grasping and bin picking. In The 13th International Conference on Soft Computing Models in Industrial and Environmental Applications 2018 Jun 6 (pp. 367-376). Springer, Cham

[5] Buchholz D. Bin-picking: new approaches for a classical problem. Springer; 2015 Nov 29.

[6] Buls E, Kadikis R, Cacurs R, Ārents J. Generation of synthetic training data for object detection in piles. InEleventh International Conference on Machine Vision (ICMV 2018) 2019 Mar 15 (Vol. 11041, p. 110411Z). International Society for Optics and Photonics.

[7] Arents J, Cacurs R, Greitans M. Integration of Computer vision and Artificial Intelligence Subsystems with Robot Operating System Based Motion Planning for Industrial Robots. Automatic Control and Computer Sciences. 2018 Sep;52(5):392-401.

[8] Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, Wheeler R, Ng AY. ROS: an open-source Robot Operating System. InICRA workshop on open-source software 2009 May 12 (Vol. 3, No. 3.2, p. 5).

[9] Zivid One technical specification 2019. Available from: https://www.zivid.com/hubfs/files/SPEC/Zivid%20One%20Plus%20Datasheet.pdf

[10] Intel RealSense Product Family D400 Series, Datasheet, 2020. Available from: https://www.intelrealsense.com/wp-content/uploads/2020/06/Intel-RealSense-D400-Series-Datasheet-June-2020.pdf

[11] ZCU102 Evaluation Board, User Guide, June 12, 2019 Available from: https://www.xilinx.com/support/documentation/boards_and_kits/zcu102/ug1182-zcu102-eval-bd.pdf

[12] Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unified, real-time object detection. InProceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 779-788).

[13] He K, Gkioxari G, Dollár P, Girshick R. Mask r-cnn. InProceedings of the IEEE international conference on computer vision 2017 (pp. 2961-2969).

[14] Cognilytica. Data engineering, preparation, and labeling for AI. 2019

[15] Roh Y, Heo G, Whang SE. A survey on data collection for machine learning: a big data-ai integration perspective. IEEE Transactions on Knowledge and Data Engineering. 2019 Oct 8.

[16] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: networked science in machine learning. SIGKDD Explorations 15(2), pp 49-60, 2013.

[17] Python Software Foundation. Python Language Reference, version 3.7. Available at http://www.python.org

[18] Community BO. Blender - a 3D modelling and rendering package [Internet]. Stichting Blender Foundation, Amsterdam; 2018. Available from: http://www.blender.org

[19] Coleman D, Sucan I. A., Chitta, S, Correll, N. Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study, Journal of Software Engineering for Robotics, 5(1):3–16, May 2014. doi: 10.6092/JOSER_2014_05_01_p3.