Deep Learning-Based Visual Control Assistant for Assembly in Industry 4.0

Mauricio-Andres Zamora-Hernandez mauricio.zamorahernandez@ucr.ac.cr[1], John Alejandro Castro-Vargas jcastro@ua.es[2], Jorge Azorin-Lopez jazorin@ua.es[2], and Jose Garcia-Rodriguez jgr@ua.es[2]

[1]University of Costa Rica, Costa Rica
[2]University of Alicante, Spain

March 16, 2021

Abstract

Product assembly process is a crucial element in factories. In the context of Industry 4.0, the offer of mass customized products is expanded, increasing the complexity in the assembling phase. This implies that operators have to pay close attention to small details, which could cause errors during the manufacture of products due to its high level of complexity. To mitigate this problem, we propose a novel architecture that evaluates the activities of an operator during manual assembly in a production cell able to identify errors in the manufacturing process. Avoiding low quality in the final product and reducing rework and waste of raw material or time. To perform the assessment, it was necessary to use state-of-the-art computer vision techniques like deep learning. So that the visual control systems can identify tools, components and actions. This work generates as contributions, a deep learning-based visual control assistant for assembly that allows to evaluate in real time the activities in the assembly process to identify errors. A language of general use to describe the actions in the assembly processes, which can also be used independently of the architecture. And finally two datasets to feed with annotated data deep methods, the first one for the recognition of tools and accessories and the second one to identify basic actions in manufacturing processes. To validate the proposal, a set of experiments have been developed obtaining high rates of accuracy.

1 Introduction

The Industry 4.0 is causing major changes in the manufacturing and service industries, by introducing Artificial Intelligence (AI) techniques into processes. Among the areas where great advances are being made, the mass manufacturing of customized products stands out. This has created new processes and quality management challenges: managing intermittent demand, sporadic production, high inventory turnover, production over long periods, or constantly creation of new products. These factors are considered to create stressful situations for operators, as they have to modify production to quickly adapt to demand.

Automatic inspection is one of the first tools (which still has many challenges) for the integration of human and machine work in the industry. It consists in the comparison of a product against the standards (a measurement of parts or components) to determine its quality. Due to the increasing need for customized products and higher levels of quality, system specifications are becoming more and more demanding. It is intended that the systems can identify, by themselves, parts of the pieces that are created for unique solutions or limited production, for which, learning becomes a fundamental aspect. Using Artificial intelligence (AI) can allow systems to make quality control decisions at workstations. According to Ferreiro and Sierra (2012), these quality inspection processes can be carried out using simple sensors (weight, color, size, etc.) (Fast-Berglund et al., 2013), or through advanced technologies such as computer vision. In this case, quality identification regarding shape or color is possible through the use of cameras (Hedelind and Jackson, 2011).

Automatic visual inspection flexibly adjusts to a diverse variety of products or processes without having to invest in new components or sensors. Moreover, it allows the intervention of operators interacting with the products and with the machines, taking into account the established mechanisms of control and information exchange (Lee et al., 2016). Techniques that apply computer vision can perform collision detection (Wang et al., 2013), navigation (Hornung et al., 2010) or augmented reality (Makris et al., 2016), among many others.

Some of the negative situations that companies are interested in mitigate include the detection and/or reduction of:

1. Incorrect executions of the assembly instructions.

2. Waste of materials or incorrect use of tools that can damage final assembly product.

3. Time lost due to lack of tools or materials that produce production line delays.

4. High Learning curve (for novice operators).

We propose a solution in the field of industrial engineering (IE), since the context of the solution is in the manufacturing cells and is related to subfields such as Work Design and Human Factors. It also incorporates elements of AI, particularly in computer vision (CV) using deep learning (DL) techniques to develop an assembly operations control scheme that through the use of CV allows corroborating that the operator follows the assembly instructions, according to the process description.

Each operator will be able to configure and control their working environment according to the description of the assembly described by a structured language. This is challenging due to the complexity of each environment due to factors such as smaller batch sizes, and shorter life cycles Patten et al. (2019). It will be very difficult for a single operator to remember all the possible combinations on how to build each product, components and their variations.

In particular, this work propose the design of an framework that allows controlling and evaluating the assembling actions of the operators in a production cell through the use of CV and a structured language for the description of assembly operations, which allows raising quality standards in manufacturing and provide a recommendation system mechanism.

This research seeks to impact any organization that needs to control manual manufacturing processes in manufacturing cells to increase the quality of their products and apply the principles of lean manufacturing to reduce waste.

Given the complexity of the proposal, the solution composed of different building blocks, where each one can be used as a general-purpose project on its own. These blocks can work in an integrated way to complete this virtual assistant for visual control. These framework key elements are listed below with a brief description (see figure 1):

- Manufacturing Cell Operations in Industry 4.0 Context

  The operator performing assembling actions is considered as the main actor and part of the system proposal.

- Manufacturing Description Language for Process Control in Industry 4.0

  It is an extensible activity description language that allows describing the assembling actions that the system will use to verify that the operator is performing the actions correctly.

- ToolSet: A Real-Synthetic Manufacturing Tools and Accessories Dataset

  A Dataset of tools and materials uses in assemblies in manufacturing cells, specifically created to train deep recognizing object neural networks.

- Video Dataset: A Real Dataset of assembling actions

  A Dataset of videos recorded in manufacturing environments that reproduce assembling actions. It has been created to train deep activity recogniting neural networks.

- Deep Learning-Based Visual Control Assistant for Assembly in Industry 4.0

  It is the core of the solution that integrates the previous components, adds the learning components, analysis of the instructions, verification of instructions with the actions detected visually and the recommendation system.

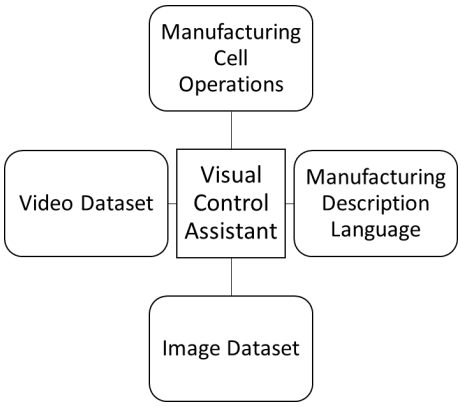


Figure 1: Proposal building blocks

The remaining of the paper is organized as follows. In section 2 we revised the state of the art of relevant related works. Next, in section 3 we present our Deep Learning-Based Visual Control Assistant for Assembly in Industry 4.0, including the visual control components, manufacturing description language and the datasets of objects and operator

actions involve in assembling tasks. In section 5 we detail the experiments developed and discuss the obtained results. To finish in section 6 with the conclusions and future lines of research.

A list of the abbreviations used in this paper is presented in Table 1.

Abbreviation	Meaning
ADV	Activity Description Vector
AI	Artificial Intelligence
CNC	Computer numerical control
CNN	Convolutional neural networks
CV	Computer Vision
DL	Deep Learning
Faster R-CNN	Faster regions with CNN
fps	Frames per Second
HSV	Hue - Saturation - Value
IE	Industrial Engineering
LIDAR	Light (Laser) Detection and Ranging
LRF	Left/Right/Frequency
LSTM	Long short-term memory
MDL	Manufacturing Description Language
MDLI	Manufacturing description language interpreter
NLP	Natural Language Processing
RGB-D	Red-Green-Blue Depth
ROC	Receiver Operating Characteristic
SSD	Single Shot MultiBox Detector
UDF	Up/Down/Frequency
YOLO	You Only Look Once

Table 1: List of the abbreviations

2 Related works

We revised research works related to the key elements of the proposal that includes: production environments, automatic inspection systems, manufacturing description languages, and deep learning-based object detection architectures and datasets, and related action recognition architectures.

2.1 Production environments

This section revised projects related to training in production environments where the use of technology can be involved to improve the training of operators. There is also an evaluation of techniques applied to automatic inspection, which is an activity to evaluate quality through the use of technology. These are key elements in the proposed solution.

2.1.1 Training in productive environments

In the fourth industrial revolution, the production model was changed by connecting technologies such as *Cloud*, *Big data*, *Analytics* and the Internet of things, allowing interaction in real-time within the same industry, as well as customers ones with producing companies Abdelhameed (2019); Zhou et al. (2018).

There is also the need to improve production tools to adapt them to new challenges in production, specially how operators must build their products. This is because the design, manufacturing, and recycling process have evolved, affecting the life cycles of products. In addition to new trends and changes in the creation of custom products that require operators to improve or learn new assembly skills.

Another element to consider is the accident rate in the industry, which varies according to the particular type of industry. Construction sectors has the higuest rate. In this case, Melzner and Bargstadt (2016) proposed to use checklists and manual descriptions to identify and reduce accidents. This proposal can be adapted to other industries where production sequences are required. Specially in those organizations where production methods have evolved into modular production schemes, which allow meeting the individual needs of users. This can lead to operator confusion when creating a large range of different products with common characteristics.

The production method has been employed from early mass production, through subsequent modular production, to today's smart production. The development of technology has made possible to meet the individual needs of users Zhou et al. (2018). Two points are considered critical in the operators training period Okubo et al. (2016):

1. A need to identify the skills that staff should have.
2. To keep the training program updated to adapt to rapid changes in techniques and technologies.

2.1.2 Automatic Inspection

One of the key goals of our proposal is the verification of the operator's activities in the assembly process. Therefore, a review of the techniques used for automatic inspection was carried out. Automatic inspection is one of the most commonly used techniques in automation. It consists of a comparison of manufactured parts or components against measurement standards to determine their quality. Due to the growing demand for personalized products with higher levels of quality, new requirements have been defined. The aim is that the systems can learn to identify parts that are created for unique or limited production solutions. Supervised learning-based systems are being used to carry out this task.

The use of AI allows an automatic inspection system to make quality control decisions. According to Ferreiro and Sierra (2012), these quality inspection processes can be carried out either by simple sensors that measure weight, color or size (Fast-Berglund et al., 2013), as well as by computer vision systems using cameras, which allow quality identification regarding the shape of products processed at other workstations (Hedelind and Jackson, 2011).

The inspections can also be made using CV- based methods since it can be applied to a diverse variety of products or processes without having to invest in new components or sensors. In this process, cooperation between operators and machines is necessary. There must be established mechanisms of control and information exchange. (Lee et al., 2016).CV is also used in manufacturing environments for quality control, collision detection (Wang et al., 2013), navigation (Hornung et al., 2010) and augmented reality applications (Makris et al., 2016).

In productive environments there are many contexts in which processes have not complete information, even they have not knowledge about relationships between its inputs and outputs. in this case, intelligent algorithms are needed to organize the information or find patterns (Ericson, Gary; Franks, Larry; Rohrer, 2016).

One key aspect in manufacturing environments is the protection of the operator's integrity; Human - Robot Interaction (HRI) can be improved by using "RGB-D" camera-based learning to track operator movements so that robots can predict intentions and recognize the behavior of the humans that work within a collaborative way (Xiao et al., 2015). This allows creating more flexible working environments to perform human tasks, but requires machine learning algorithms to make decisions based on datasets of partially known environments, having to establish control policies for the protection of people. Santoro et al. (2008) proposed to use a mixed scheme, using supervised and unsupervised learning where the datasets used to train behavior patterns have been annotated by a human "trainer".

2.2 Description of manufacturing processes

We reviewed works that use languages to represent the necessary instructions for the assembly of products by the operators. The use of languages is required to express the instructions to be performed in a formal way. For this purpose, it is required to defined a particular semantics of each language.

IBM proposed in the mid-1980s a language for the product assembly process. However, due to the context of that time, it was assumed that only robots were used. It was designed for CNC-type machines that performed repetitive actions of exact positions, without any kind of cognitive analysis (Nackman et al., 1986).

We identified several works that use CV systems to identify actions developed in the scene and generate descriptive text to represent them. In some cases, proposals use Natural Language Processing (NLP) techniques; but since they do not have semantics, they do not generate a standardized format. For this reason, that would complicate the process of analyzing manufacturing-oriented actions.

One of the most advanced challenges in DL is the understading of actions made by a human (in this case an operator) through CV, so that a machine performs the same actions as an operator human through an observation process (Real et al., 2019). One of the ways to teach DL-based system to recognize actions is through a task detection technique that can generate captioning output in sentences using natural language, but without grammar, because this form of language lacks semantics. Consequently, that does not allow to formaly express the operator behaviour (Real et al., 2019).

Yang et al. (2015) developed systems using probabilistic representations of modulation-based grammar manipulation that parse instructions to visually represent sentences to manipulate robots. To mitigate the aforementioned deficiencies, they proposed a system of convolutional neural networks, which, through video analysis constructs grammar trees of the observed actions. These are non-restricted general use grammars for a particular use. So, it could generate unstructured actions for the strict verification of what is captured. In this line, (Mancini et al., 2018) already worked on object detection in specific domains of industry, but without the definition of grammar to describe the sequence of actions. For these reasons, the proposal of creating an assembly specific-domain grammar in Industry 4.0 is a novel and necessary tool.

2.3 Images and Actions Datasets

One of the crucial elements to complete this investigation is to properly carry out the deep models training. We need that our system recognize an locate tools, accessories, and parts present in the working area to recognize and disambiguate the actions performed by the operators. For this purpose, we looked for images and actions datasets for the detection of objects and actions manufacturing cell recorded video sequences. In the review, it was not possible to identify an adequate database of images or actions that fitted our requirement for the context of the problem. The following are the general context dataset of images and actions found.

2.3.1 Image Datasets

We made a review of the public datasets that fitted to the needs of this work. None of the datasets found fitted our requirements. Since these datasets contained a limited amount of tool images, views were not consistent for training and the datasets did not have the required tool classes. This was one of the reasons for the creation of the customized dataset that is described in section 4.1.

We revised the most relevant public general purpose datasets for objects recognition. Althought these datasets do not meet the image characteristics required for this project, they provided images that can be used to add backgrounds for data augmentation.

The Pascal Visual Object Classes (VOC)[1] is a public dataset and an annual competition along with workshops since 2006. The dataset consists of 500,000 images divided into 20 categories that were retrieved from flickr[2] (Everingham et al., 2014).

ImageNet Large Scale Visual Recognition Challenge[3] contains 14,197,122 images, 21841 synsets indexed, organized according to the WordNet hierarchy. The challenge has been run annually since 2010 and it has become the standard benchmark for large-scale object recognition. The dataset contains a set of manually annotated training images (Russakovsky et al., 2015).

COCO Dataset (Common Object in Context)[4] is a large-scale object detection, segmentation, and captioning dataset. It contains 330K photos of 91 objects types with a total of 2.5 million labeled instances. It has considerably more object instances per image as compared to ImageNet and PASCAL VOC (5 captions per image) (Lin et al., 2014).

2.3.2 Action datasets

In this section we present relevant action recognition datasets, as well as a summary that highlights important information about each of them. The reviewed datasets have been selected based on their innovation, types of actions they include, how the data is acquired and its scale, which has to be large enough to adequately train deep networks.

RGBD-HuDaAct[5] This is a dataset specialized in the recognition of human activity in domestic environments. It consists of 12 classes and more than 1,000 videos, with a total of almost 46 hours of video. Classes cover actions ranging from making a phone call, taking off your jacket, cleaning the floor, and eating or drinking water. It uses information from low-cost *RGB-D* cameras and depth sensors, Ni et al. (2011a,b).

UTKinect-Action3D[6] This is a dataset with *RGB-D* information and skeletal joint locations. It contains 10 types of recorded actions, including walking, sitting, getting up, lifting, carrying, throwing, pushing, pulling, shaking hands, clapping. The dataset was recorded by 10 different users, each user performed each action twice. The technical characteristics of the videos include: frame rate is 30 fps, *RGB* images have a resolution of 480x640 and the resolution of depth images is 320x240, Xia et al. (2012).

UCF101[7] This dataset, created by the Central Florida University [8], is made up of videos obtained from *Youtube* [9]. It contains 101 actions and 13,320 short videos. They are classified into five categories: human-object interaction, body movement, human-human interaction, playing musical instruments and sports. Among the technical characteristics: they were recorded in unrestricted environments, including camera shake, various lighting conditions, partial occlusion, poor quality *frame*, etc. All videos have a fixed a *framerate* of 25fps and a resolution of 320x240. The video format is AVI compressed using the DivX codec, Soomro et al. (2012a,b).

2.4 2D Objects Detection Deep Architectures

We revised two of the most used deep architectures for object detection: You Only Look Once (YOLO) and Single Shot MultiBox Detector (SSD). Based on the analysis of both architectures, it was determined the use of YOLO due to its very high levels of accuracy. In addition, it has wide support from the community of programmers and researchers.

[1] http://host.robots.ox.ac.uk/pascal/VOC/
[2] https://www.flickr.com/
[3] http://www.image-net.org/
[4] http://cocodataset.org/
[5] https://publish.illinois.edu/multimodalvisualanalytics/rgb-d-human-activity-recognition-and-video-database/
[6] http://cvrc.ece.utexas.edu/KinectDatasets/HOJ3D.html
[7] https://www.crcv.ucf.edu/data/UCF101.php
[8] www.ucf.edu
[9] www.youtube.com

2.4.1 You Only Look Once

YOLO is an architecture specialized in the detection and tracking of multiple objects in real time, generating coordinates for each object. To achieve this high-speed detection, the precision is reduced, although the method retains very high levels of accuracy (Aggarwal, 2018; Lv et al., 2020). This capacity can be used to perform actions such as autonomous driving without specialized sensors (Redmon et al., 2016).

YOLO Architectures are includes end-to-end network structures. This type of network structure is more concise compared to the two-stage networks of R-CNN type structures, which first generate recommended candidate regions and perform detection and localization. It integrates the detection mechanisms of the candidate area, making the network faster than its counterparts of R-CNN architectures (Xu et al., 2020).

The network that forms the backbone of YOLO is based on Darknet-53 to extract features from images. The entire network uses five residual layers with different scales and weights as building blocks (Li et al., 2019). The convolutional layer uses alternate convolutional kernels of 1 x 1 and 3 x 3 to extract more abstract features (Cao et al., 2019).

The anchor box concept, introduced by "Faster R-CNN" and k-means, was adopted by YOLO v3 to determine the radius size for the anchor box which locates the searched object. The parameters are relative to the predicted anchor box, instead of directly mapping the coordinates into the bounding box (Li et al., 2019).

2.4.2 Single Shot MultiBox Detector

It is a method of single-shot detection for multiple categories, where the core of the model is to predict category scores and scroll boxes for default bounding box groups using small convolutional filters to be applied feature maps. This method improves detection accuracy, in addition to improving detection speed (Liu et al., 2016; Gong and Shu, 2020).

The high detection precision in SSD is achieved by using multiple boxes or filters with different sizes and aspect ratio for object detection (Khandelwal, 2019). SSD has two components:

1. A backbone model: it is generally an image classification network, previously trained as a feature extractor. This is typically an ImageNet-trained network like ResNet from which the fully connected final clasification layer has been removed (ESRI, 2020).

2. A header: it is composed by one or more convolutional layers added to the backbone.The outputs are interpreted as the bounding boxes and object classes in the spatial location of the activations of the final layers.

SSD takes advantage of Faster-R-CNN, YOLO and multi-scale pyramids. This discretizes the output space of bounding boxes into a set of default boxes, each feature map has its aspect ratio and scale. Additionally, this model combines predictions for multiple feature maps with different resolutions to naturally handle objects of various sizes (Tang et al., 2020).

2.5 Action Recognition

Recognition of actions is a key element for this research work since one of the crucial factors lies in identifying what actions an operator is performing. In this way, it will be possible to assess whether assembling is performed according to the specifications that determine how to manufacture a product.

Yang et al. (2015) clasify human action related research into two basic areas: recognition and prediction of actions. The first is capable of recognizing the actions carried out by a subject in a video that contains the complete action. The second is capable of predicting actions from temporarily incomplete video data. These works have a wide diversity of uses in the areas of video surveillance, ambient assisted living, smart shopping spaces, etc. Abellan-Abenza et al. (2019).

Depending on the type of visual input, it is possible to divide this field into 2D and 3D image-based approches. Since 2D images introduce ambiguity to recognize some types of actions, it is becoming popular the use 3D cameras (*RGB-D*, *LIDAR*, etc.) (Park and Kim, 2019) .

Yan et al. (2019) uses the recognition of actions to determine relevant elements in the study of the work method, such as work posture, attention to work, and mental fatigue. Although this research focuses on the identification and monitoring of actions within an assembly process, elements that could negatively affect performance in the tasks carried out are not considered.

InYang et al. (2019), the detection of actions developed by humans is based on two elements: the extraction of the action features, which normally uses human biometric characteristics, and the movement detection, which seeks to identify the limits to obtain the induced skeleton with two indicators: the position of the body and the circular movement of the body segments. Most of these works used two-dimensional data. However, thanks to the recent improvement in techniques and available *hardware*, works such as Park and Kim (2019) propose the use of 3-dimensional information to improve the recognition of actions. On the other hand, authors such as Abellan-Abenza et al. (2019) argue that by using different combinations of movement characteristics, significant improvements can be obtained in the recognition of actions. Kong and Fu (2018) proposes to merge multiple characteristics extracted from videos for the recognition of actions in an effective way.

A related technique that could help action detection is *Image Captioning*. Captioning generates an automatic textual description of an image Luo et al. (2019). This description can be used to describe what happens in assembly

environments, for example, checking the necessary steps in the manufacturing process. But in general, the description with a simple label is not enough and we require the complete description of performed actions. Therefore, researchers such as Wang. Et al. Wang et al. (2018), have developed work on *captioning* for video, where they use *Hierarchical Reinforcement Learning* techniques to generate descriptions. Krishna et al. Krishna et al. (2017), are using *long short-term memory* (LSTM) techniques in a *dense-captioning* model of event detection. In both cases, descriptive narratives in natural language are used. In robotics, there are works that teach robots to perform the same actions as a human operator, just by observing it. In this line, Nguyen et al. Real et al. (2019) proposed to understand and imitate human actions without defining objectives or validations in the actions.

3 Visual Control Assistant for Assembly Architecture (VCA^2)

In this section, we describe our Visual Control Assistant for Assembly Architecture (VCA^2) proposal. First, the overview of the architecture is presented to show the modules and relationships. Finally, each of the modules is explained in detail in the following sections.

The architecture (see figure 2) is made up of three modules that are detailed in the following sections: visual processing, manufacturing description language processing, and the action command processor.

- Visual processing is the essential module of the architecture that allows perceiving the working environment and the actions carried out by the operator to form the action commands. These commands jointly specify the action carried out and the composition (tools and components) of the detected environment (see Section 3.1).
- The module of the manufacturing description language processing enables to interpret the assembly instructions to decide if what the operator does is correct.

  Also, the relevant characteristics of the proposed language used to describe manufacturing actions are explained in Section 3.2.
- Action command processor is the module responsible for taking the inputs from the previous stages (the visual processing and manufacturing description language processing) to determine if the operator is performing the actions according to the description. Furthermore, it has the responsibility of informing if the operator is performing the operation properly or has made a mistake. Moreover, it will be able to recommend actions to the operator (see Section 3.3).

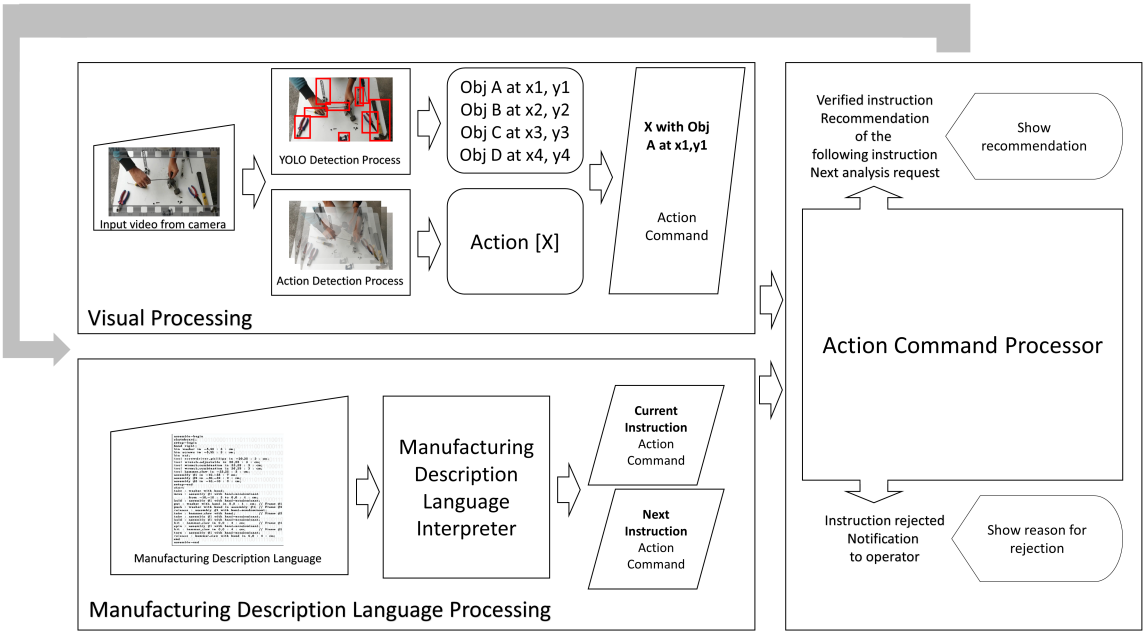


Figure 2: Overview of the proposed architecture VCA^2

3.1 Visual Processing (VP)

This module is in charge of performing a visual inspection of the operator and the environment while assembling the parts. To do so, it monitors both the tool and components used and the actions to carry out the assembly. The module is composed of two sequential units that are detailed below: YOLO Detection Process and Action Detection Process. The outputs of these modules are combined to provide the Action Command, which will be used later to evaluate if what the operator is doing is the instructions as defined by the language (see Section 3.2.1).

3.1.1 YOLO Detection Process

The main function of this module to detect tools and pieces (e.g. nuts, bolts) and their location in the image. Although the proposed architecture is generic and any object detector could be used, in this paper the module is based on YOLO (see section 2.4.1), which provides real-time detection with high accuracy performance. A specialized dataset was designed with images of tools and parts, commonly used in the assembly processes, to carry out the learning phase. An example of the elements detected by the YOLO Detector can be seen in figure 3.

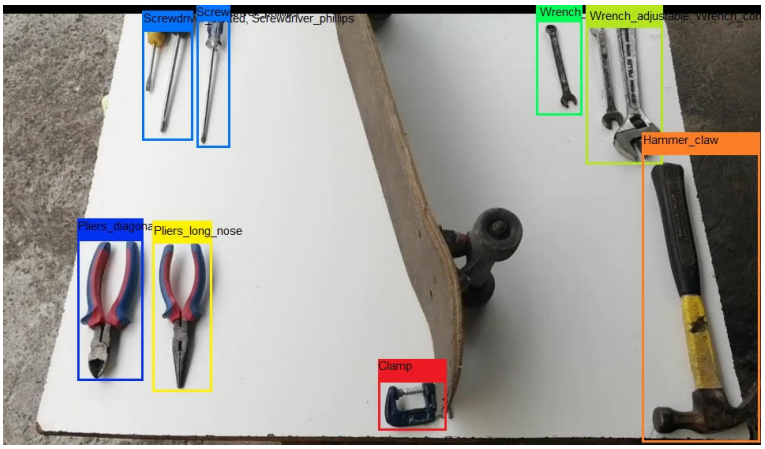


Figure 3: Example of manufacturing scene with marked objects detected by YOLO

3.1.2 Action Detection Process

Since VCA^2 could be mainly analyzed as an instruction recognition architecture, it must be able to identify what action the operator is performing to validate the adequate assembly or to propose the next step to carry out. This module provides this information by analyzing the sequence of video frames that come from the video capture module.

This module will generate the action as output, which combined with the object detector output will form the Action Command that will be used as input for the next stage (the Action Command Processor). The action processor engine is based on the Deep Activity Description Vector (D-ADV), a descriptor of image sequences that can describe with high accuracy the actions or activities that are being performed on it (Borja-Borja, 2020). This descriptor is the variant of the Activity Description Vector (ADV) (Azorin-Lopez et al., 2016) that can be used for deep learning methods.

The ADV and its variants have been successfully applied in different areas for the detection of actions of individuals and groups by analyzing the movement of the person as a whole. In this case, the descriptor is used to detect actions observing the working area of a production cell. The system is only able to see the tools, parts, components, and hands of the operator. The D-ADV can be used with different deep network architectures. In this paper, a two-stream architecture has been used as shown in Figure 4.

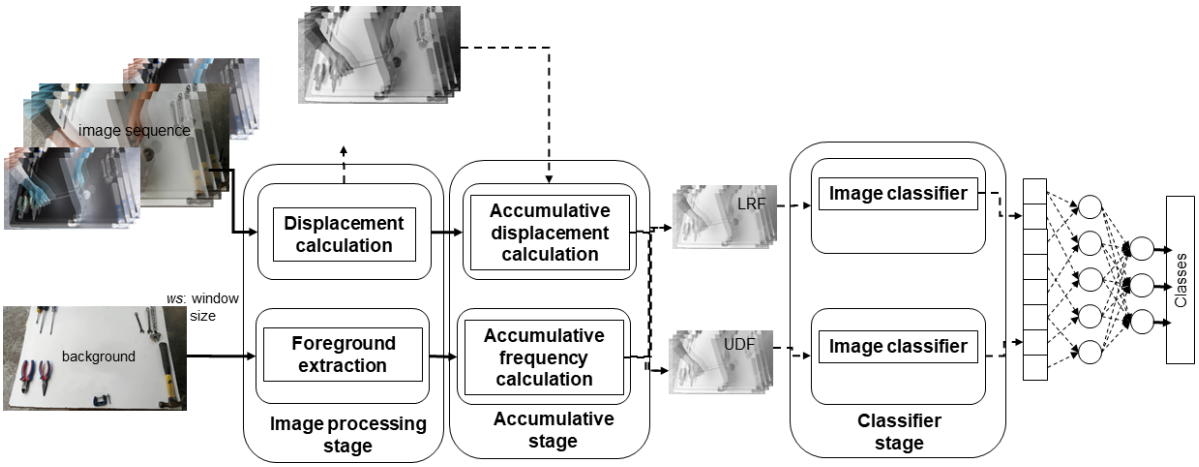


Figure 4: Deep network architecture for action detection

As seen in Figure 4, using the video that comes from the working area of the operator, the architecture **is able to calcute the action carried out by the operator in the sequence. In order to provide the corresponding class to the action, three stages are involved in the process: image processing, accumulative and classifier stages. The first two stages (image processing and accumulative stages) proceeds with the**

calculation of the image sequence representation of the deep variant of the original ADV. For the first stage, the movement ocurred between two consequtive frames is calculated by means of the optical flow algorithm. The movement is separated into displacements carried out for each axis (positive or negative) considering the up (U), down (D), left (L), right (R) movements with respect to a local origin of coordinates. At the same time, the foreground of the scene is estimated to calculate the frequency (F) as the number of movements occurred in a specific region of the scene (this method divides the scenario into regions of grid cells to discretize the environment). Once the displacements and the frequency are calculated, they are accumulated according to a determined number of frames specified as the *window size* (WS). Finally, these individual movements and frequency are concatenated in D-ADV to define two images LRF (Left/Right/Frequency) and UDF (Up/Down/Frequency) which are the input queue of the classifier stage. Therefore, the D-ADV is able to compactly represent an action sequence by means of only two images. To calculate them, the input frame and the inmediately WS previous frames are considered.

The classifier stage is a two-stream neural network composed of Convolutional Neural Networks (CNN) able to classify the input into many classes. The individual streams are connected at the end of the network via late fusion providing the class of the action.

3.2 Manufacturing description language processing

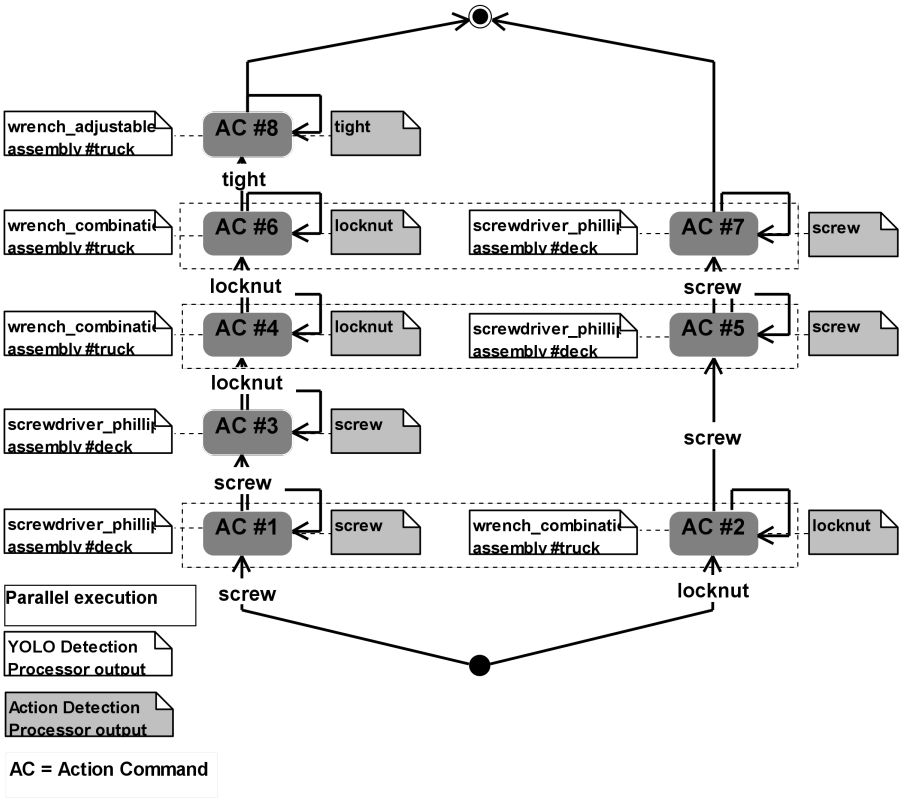


Figure 5: Example of state graph for manufacturing description language processing

This module generates a state graph with the actions, including the tools to be used, that the operator must perform (see Fig. 5. The language was designed specifically to describe operator actions (for more details see section 5.2). An example of the code can be seen in the listing 1. Consequently, it processes the assembly sequence with the instruction language, generating the expected outputs. Both the state graph and the result of the visual processing are input to evaluate the operations carried out in the working area.

Specifically, the manufacturing description language interpreter (MDLI) interprets the instructions described by the manufacturing description language to generate, as a result of the interpretation process, the state graph with the evaluated steps. Each node represents a state of the assembly and the edge, the trigger for each state change through the actions and tools to advance to the assembly process (see Fig. 6).

This graph is the one that will be used to compare the output generated from this module and that generated in the visual processing. It will allow determining if the instructions (actions and tools) were performed correctly, keeping track of the last execution carried out. With this analysis, it is possible to determine which is the current instruction to execute and the next instruction to be used as a recommendation. The interpreter generates the instructions in the format of Action Command (AC).

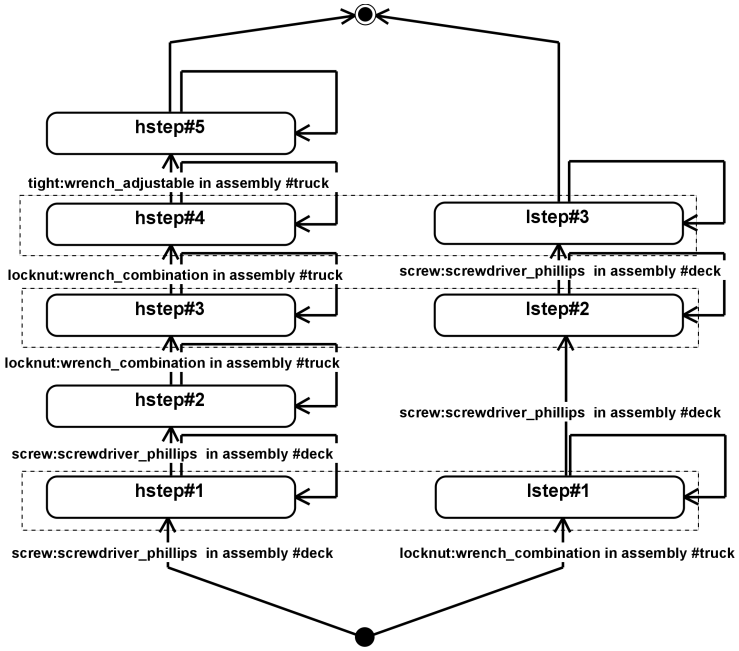


Figure 6: State diagram representing the manufacturing language that will be interpreted by the action command processor

3.2.1 Manufacturing Description Language for Process Control in Industry 4.0 (MDL)

The Manufacturing Description Language for Process Control in Industry 4.0 is a language that has been designed to specify the necessary actions that the operator has to perform in the assembly process. This language has the possibility of being used independently.

During the design of the language, it was established that it is a basic part of a global solution, which would use AI systems, to indicate the general operation settings automatically, such as the restrictions of the working area, tools to be used, parts and components necessary for the assembly. Anyway, for environments where this information is not available, the language provides a setup section, where the parameters can be set manually.

Another consideration of the language, during the design phase, was the syntactic structure of the instructions, which had to be simple and concise, requiring brief training in their use. Specially, since it can be used by people of different specialties, such as quality or production managers, or operators of the production cells.

Among the relevant characteristics to highlight: the method allows us to define an assembly by components, being an advantage because a milestone can be established during construction, for quality checks during assembly. It also allows establishing alternative assembly routes.

A basic set of tools was defined, but the system gives the possibility to extend with new configurations. The currently available tools are:

- Clamp
- Gun Drill
- Screw Drill
- Ball Pein Hammer
- Claw Hammer
- Nut Driver
- Diagonal Pliers
- Lineman Pliers
- Locking Pliers
- Long Nose Pliers
- Ratchet
- Electric Screwdriver
- Phillips Screwdriver
- Slotted Screwdriver
- Socket
- Adjustable Wrench
- Allen Wrench
- Combination Wrench

Also as a complement to the tools, some have accessories for which they were defined in the language, as well as parts and elements of general use, among these are:

- Bolt
- Bit Drill
- Gears
- Nut
- Screw
- Washer

Figure 7: Frames for code example

Listing 1: Code example

```
assemble-begin
skateboard;
setup-begin
hand right;
bin washer in -5,20 : 3 : cm;
bin screws in -5,25 : 3 : cm;
bin nut;
tool screwdriver_phillips in -30,35 : 3 : cm;
tool wrench_adjustable in 30,35 : 3 : cm;
tool wrench_combination in 33,35 : 3 : cm;
tool wrench_combination in 36,35 : 3 : cm;
tool hammer_claw in -33,35 : 3 : cm;
assembly #1 in -10,-35 : 7 cm;
assembly #2 in -30,-32 : 2 : cm;
assembly #3 in -10,-10 : 2 : cm;
setup-end
start
take : washer with hand;
move : assembly #1 with hand-nondominant
from -10,-10 : 3 to 0,0 : 4 : cm;
hold : assembly #1 with hand-nondominant;
put : washer with hand in 0,0 : 1 : cm; // Frame #1
push : washer with hand in assembly #1; // Frame #2
release : assembly #1 with hand-nondominant;
take : hammer_claw with hand;              // Frame #3
take : assembly #1 with hand-nondominant;
hold : assembly #1 with hand-nondominant;
hit : hammer_claw in 0,0 : 4 : cm;         // Frame #4
spin : assembly #1 with hand-nondominant;
hit : hammer_claw in 0,0 : 4 : cm;         // Frame #5
turn : assembly #1 with hand-nondominant;
release : hammer_claw with hand in 0,0 : 4 : cm;
end
assemble-end
```

3.3 Action command processor

This is the last stage of the architecture and allows us to determine if the assembly is being carried out correctly or to recommend the next action to take into account to assemble a product.

The module has as inputs the Action Command (AC) (object detected in the scene and the action performed by the operator) calculated from the visual processing module and the state graph of the manufacturing description language processing. The module processes the state graph according to the *action command* for each frame of the video. In other words, the graph is crossed through the actions performed by the operators and tools in the scene *frame* to *frame*. To avoid outliers and specific errors of the visual processing module, the system changes the state if the change to a new state is based on the labeling of a set of *frames* that provide the same result (i.e. or the mode through a window).

Therefore, if the system is designed to validate whether an assembly is adequate, the state graph must be traversed from the initial node to the final node when the operator follows the instructions according to the specifications. If the operator passes through all states, the assembly will be considered to have been satisfactorily executed. The method will evaluate the operator's actions in real-time, each action is equivalent to a state of the graph, when making the transition from a valid state (recognized action) to another valid state, the system will indicate to the operator that he or she is making the assembly correctly. If the action is not recognized by the system during the evaluation, it is considered to be an invalid state, and the system will wait some seconds (to be set in the system) until the transition to a valid state occurs. If not, it generates an alert to the user and blocks the execution graph. If the operator performs an action that is not planned in the state graph, the system will hang at any node, providing an alert after a set number of seconds (to be set in the system). In case the system is designed to recommend actions, the system will provide the different actions associated with the edges of each visited node.

4 Toolset and Manufacturing Video Actions Datasets

Since we do not find specialized datasets in assembling processes, we created our own datasets to train with annotated data our Deep Learning-Based Visual Control Assistant for Assembly in Industry 4.0. The first dataset is composed of static images of typical tools and elements in cells, created with real images using data augmentation techniques and additional generation of synthetic images. These images are necessary to train the YOLO network. The second dataset consists of assembling actions videos necessary to train the action detector.

Image and action datasets are available in four projects on Github. Due to the maximum limit allowed by each project. To access the datasets you can go to Github [10].

[10] `https://github.com/mazamorahdez/Dataset-for-Visual-Control-Assistant-for-Assembly-in-Industry`

4.1 ToolSet Dataset

Toolset was specifically created to satisfy our research specifications and based on the details of the proposed language called Manufacturing Description Language for Process Control in Industry 4.0 (see section 3.2.1). So, the categories of the tools are shared by both: the language and the dataset.

Toolset is an hybrid dataset consisting of 24 categories and 80,000 images including real and augmented ones. The elements contained in the dataset are general-purpose tools for manufacturing processes, tool accessories and general-purpose parts for assembly, such as screws, nuts, washers, and others.

The specifications of this dataset were designed to be trained with YOLO (see section 2.4.1), so labeling was done for this purpose, both for real pictures and for synthetic augmented images.

(a) Examples of Real augmented images

(b) Examples of syntetic images

Figure 8: Examples from ToolSet Dataset

To collect the basic (real) images, we searched for each category using the datasets described in the object recognition datasets section (see section 2.3.1). But, since we did not find enough images for the defined categories, a web search was done using Google Images® [11] obtaining a base of 1000 real images free of use, with an intermediate to high resolution, which was later used to apply data augmentation techniques.

Regarding the data augmentation process, transformation, translation and background change techniques were applied. A total of 50 transformations were performed for each object, using 1000 images to establish random backgrounds. The generation of real data datasets takes a lot of work, since they have to search and manually tag each one of the images. To increase the size of the dataset, computational techniques were used to generate synthetic images from 3D models. To obtain a greater variety of images that would satisfy the requirements for deep network training. To generate synthetic images we used UnRealEngine, a well-known video game engine, together with a plugin called UnrealROX Martinez-Gonzalez et al. (2020), that facilitates the task of creating the synthetic dataset, allowing the creation of images from different points of view. Moreover, we performed the labeling process for the YOLO format.

4.2 Video Action Dataset

To carry out the training of the action recognition architecture, we recorded several videos of manufacturing tasks to carry out the training. The camera was located on the top of the centroid of the workstation. 37 videos were recorded, from which 18 shots were obtained on average for each of the 12 actions. Figure 9 shows an example of some of the frames from one of the training videos.

The training videos were recorded in domestic and lab environments. Furthermore, recorded tools are in some cases deteriorated by use, having characteristics that usually generate noise such as wear and light oil stains. In this way, we validate detection in different environments. The videos were taken with HD web cameras, optical glass lenses, with autotracking and autofocus technologies, located one meter away from the work area.

The videos were made with simple assemblies of common elements. The assembly of a skateboard was chosen, for which different techniques and tools were used to carry out the tests. Considering different positions of the operator and different lighting levels.

To add variability to the videos, in addition to the changes made in the previous point, the sequences that describe how to build the skateboard were modified, since the assembly can be carried out in different configurations. Different positions of tools and parts within the work area were also applied.

[11] https://images.google.com/

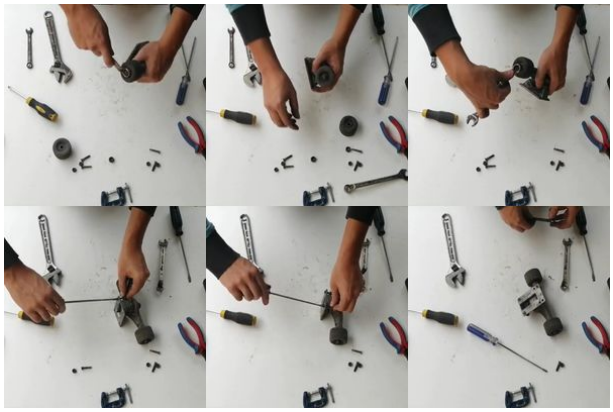


Figure 9: Examples of video dataset

5 Experiments

A set of experiments has been developed to test the functionality of our system. We used some examples of manufacturing processes that, on one hand, have been translated to the proposed manufacturing description language and converted in a compact sequence of actions to be performed by the operator. On the other hand, a set of videos were recorded and annotated for the proposed dataset. The videos were processed by the visual processing module to detect and recognize tools and components, and to recognize the list of actions developed by the operators at different frames. Finally, the action command processor is used to compared both, the sequence of actions described in the state graph and the representation obtained from our deep video processing architecture.

The experiments proposed in this research were performed on two twins workstations with Ubuntu 18.04 LTS, Intel (R) Core (TM) i5-8400 2.8GHz 6 cores 9Mb cache, 32GB RAM, GeForce RTX (TM) 2070 with 8GB GDDR6 256-bit memory interface, two 480GB solid hard drives. The TensorFlow 2 (which includes Keras) was used as the deep learning framework.

5.1 Description of the validation videos

The videos used for the validation were recorded in a working environment with natural lighting, HD web cameras, optical lenses with auto-focus technologies. The camera was located in the center of the manufacturing cell workbench. The videos were recorded with operators with different characteristics, including skin color, height, wearing long-sleeved and short-sleeved clothes, and in different positions on the workbench. These to generate variability to test the recognition of actions. Moreover, the lighting conditions are very different for each sequence of the dataset.

The videos show the construction of a component for a skateboard, where the pieces and tools are arranged on the surface of the workbench. They have different variations until completing the final assembly. The operator does not have a full representation of his body, only both hands are recorded while the assembly is executed.

An example with videos containing the actions (Table 2) that were selected from the language elements can be seen in figure 10. In these images, the setup characteristics of the camera, as well as the working area, are shown: point of view, type of tools that the system can detect, positions of the operators' hands, etc.

ID	Action	Sequences	Frames
a	tight	179	4784
b	locknut	30	8121
c	screw	47	7879
d	hold	34	3457
e	hit	46	427
f	drilling	25	741
g	release	24	979
h	put	26	1520

Table 2: **Actions considered in the experiments, number of diffent sequences for each action and frames containing the action.**

5.2 Manufacturing Description Language experiments

An experiment was designed with a group of industries that use manufacturing cells to carry out the manufacturing language validation. A representative of each industry participated in an interview that aimed to identify how the

(a) Example of wrench tight

(b) Example of wrench locknut

(c) Example of screwdriver screw

(d) Example of pliers hold

(e) Example of hammer hit

(f) Example of drill drilling

(g) Example of ratchet release

(h) Example of ratchet put

Figure 10: Examples of actions with tools

instructions are represented during the assembly process. In addition, the tools they normally used to carry out these tasks were described.

For the design of the experiment, two phases were established to carry out the validation, which aimed to generate a language that was conceptually equivalent to the regular instructions of each factory.

The first phase of the experiments consisted of:

1. Identify examples of production sequences by each industry to generate the verification.

   During the research, the different ways of how the companies document their processes and how they assemble their products were compiled. This information was taken and the equivalents of the instructions of each company were generated using the Manufacturing Description Language.

2. For each company sequence, a translation of the instructions into the created language was carried out to assess the level of equivalence between both.

   With the help of counterparts in the companies participating in the experiment, the instructions generated with the original versions were reviewed to determine the level of accuracy. It was possible to determine that approximately 97% of the instructions were able to convert, the remaining 3% correspond to very particular elements of each industry. To solve this point, we proceeded to generate an extension of the language. It is important to indicate that this was a consideration that was taken as a requirement when creating the language that was extensible with the incorporation of new tools and actions.

3. A test was carried out taking a group of operators having a brief introduction on the use of the language to execute it on the workstations.

After verifying, in the previous phase, that the language could be used to describe an assembly process, the second phase of language verification was established taken complete assembly processes. Several videos were recorded by the research team and generated the complete code of the instructions to reproduce the processes.

5.3 Visual processing experiments

In the experiments for object detection, YOLO was selected as the basic architecture of the network for inference of objects, because it has a high performance in detecting objects in real-time as can be seen in the characteristics described in the section 2.4.1. A subset of the objects available in the dataset was used for training to carry out the experiments. A maximum of 50,200 epochs is set during training. 18 of the 24 classes of objects were used, including hand tools such as screwdrivers and hammers.

Real data were mixed with synthetic one in a 20/80 ratio to generate a training and validation set. Moreover, data augmentation was applied including rotations of up to 40 degrees, variations in HSV of up to 50 %, and scale changes of up to 30 %, starting from an image size of 416. The hyperparameters used to train the network were a learning rate of 0.001, momentum 0.9, and a burn-in of 1000.

An example of the identification of the objects in one of the validation videos is shown in Figure 3, where the bounding boxes of the tools used, for each of them the class identification correspondent can be seen.

Regarding the validation of the Action Recognition Processor, **a Resnet50 architecture has been considered as the base of each stream for the classifier stage and a window size (WS) of 10 frames to accumulate the displacements to conform the images (see Sect. 3.1.2). Regarding the training, a transfer learning approach consisting on three stages has been considered. First of all, the last layers of the Resnet50 network trained on the ImageNet have been replaced by fully connected layers that are fine tuned with the new classes correponding to the actions. Moreover, two fully connected layers placed for late fusion of both stream are also trained. Only the top layers of Resnet50 (one for each stream) and the corresponding for late fusion are trained with a learning rate of** 0.0001**. For the second stage training, only the 22 bottom layers of each Resnet50 are trained because of the input of the** LRF **and** UDF **images are from a different domain of the RGB images of ImageNet. Finally, the 30 top layers are fine tuned again to provide the last adjustment.**

The eight basic actions considered in Table 2 were evaluated since they were taken as the basic core described in the language. In order to do this, a 3-fold cross-validation for sequences was used to select the train and test sets to validate the proposal. The results consider two criteria: *Frame* and *Sequence* performance. *Frame* performance considers the results of the action recognition based on each single frame from those used in tests whereas the *Sequence* performance considers each sequence (see in Tab. 2) as a unit, this is the whole sequence is labelled as the most predicted action for the frames in it. The results can be seen in Table 3. Regarding *Frame*, the overall performance for recognizing actions is very high in terms of the sensitivity and specificity (96.37% and 97.9% respectively.For the specific classes, there are actions that are very different from the others, such as drilling and hit that 100% of sensitivity were obtained during the execution of the tests. There are other actions whose execution with the tools are very similar, but they have slight variations regarding the way in which the operator performs the action. These actions are: tight, put, release that has approximate value to 98% of sensitivity. For the eight actions evaluated, five actions achieve a sensitivity greater than 98%, which the language considers to have very high performance when recognizing the actions. For the remaining three actions (hold, locknut, and screw), sensitivity values were 94.65%, 5.28% and 96.47% providing again very high-performance values. **Finally, regarding *Sequence*, the performance for both sentivity and specificity are very high achieving 99.36% and 99.42% in overall. Actions as screw, hit, drilling, put and release are perfectly recognised without having any false alarm.**

Figure 11 shows an analysis of these data using the Receiver Operating Characteristic (ROC) curve. These evaluation metrics allow us to verify the performance of the classification model graphically. As can be seen in the values of both the analysis of the complete sequences and of each frame, practically there are values very close to 1 (maximum value) as true positive, which indicates that the tests carried out have high levels of confidence in terms of recognition of action.

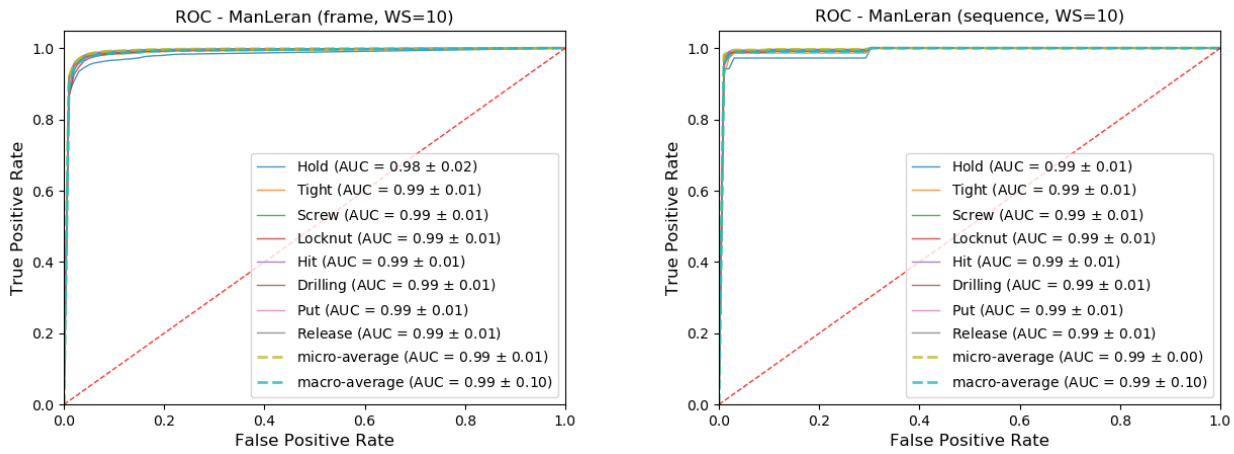
Class	**Frame**		**Sequence**	
	Sensitivity	**Specificity**	**Sensitivity**	**Specificity**
Hold	94,65 %	94,83 %	97,06 %	99,19 %
Tight	98,79 %	98,76 %	99,44 %	99,12 %
Screw	96,47 %	96,45 %	100,00 %	100,00 %
Locknut	95,28 %	95,18 %	98,33 %	96,82 %
Hit	100,00 %	100,00 %	100,00 %	100,00 %
Drilling	100,00 %	99,99 %	100,00 %	100,00 %
Put	97,89 %	97,88 %	100,00 %	100,00 %
Release	98,06 %	98,14 %	100,00 %	100,00 %
Overall	**96,37 %**	**97,94 %**	**99,36 %**	**99,42 %**

Table 3: Results of action recognition

5.4 Action command processor experiments

The Action command processor was evaluated by applying the principles of a parser since it verifies that the user's actions (coming from the Action Recognition Core) in the form of commands are considered as tokens that evaluate the grammar of a language. It is important to highlight that a program in this language is the sequence of instructions described by the quality or production engineers and written using the Manufacturing Description Language.

In this first part of the action command, the test videos were used and a table with the identified actions generated.



(a) ROC results from Frames

(b) ROC results from Sequences

Figure 11: ROC results

This is the input together with the tokenized instructions to generate a state diagram, which is in charge of verifying that the transition between the states (a frame containing a recognized action is considered as a state). The video is reviewed in real-time to determine at which point the state changes. This state is compared with the state diagram of the instructions. If the transition is fulfilled, the process is considered to be "correct execution". The process ends when all the state transitions are completed according to the definitions until reaching the final node, the execution condition is changed to "complete execution".

All the videos tested, successfully generated the state diagrams. And therefore it was possible to generate the "complete execution" condition. The state diagram allowed us to establish which state changes were valid to make the recommendations.

In the figure 12 a representation of a process is shown, it was generated from one of the validation videos. This graph has two sections, the upper part shows the graph that is defined by the MDL instructions (the white blocks are the actions not executed and the dark ones are the actions that have already been executed). At the bottom, there is a timeline of the execution of the actions that are identified by the system, in red color is the action that is actually happening in the video and in blue color what the system identified. This is also relevant because the level of precision of detection by the system can be appreciated against reality.

The objective of the figure is to compare how the graph completes the execution of the actions in synchrony with the detection of actions by the system. In this example it can be seen that the Action Recognition Processor (ARP) in the action of the screw, detects a little earlier than the video is labeled as a screw. This is because the video is considered as a screw until the moment you turn the screwdriver, the ARP determines it from the moment the screwdriver tip touches the screw.

6 Conclusions

This paper presents a novel proposal to solve the prevailing problem in manufacturing for Industry 4.0 consisting of monitoring an operator during the manufacturing of a product or assembly. The proposed architecture of a visual control assistant based on Deep Learning, determines that the actions are carried out as established by the managers (quality or production engineers). Hence, the architecture is in charge of the assembly verification process and can provide systems based on it to indicate if the process was completed correctly or to alert if the steps carried out by the operator were not following the instructions. Direct benefits are obtained for stakeholders by performing this automatic assembly check: the operator improves his productivity rate, especially in those cases that are novels or are learning the assembly of a new product, since the system can provide them with confidence; the factory improves its production capacity, reduces costs for product loss due to damage or development failures, gets shorter delivery times to customers; and, to finalize, the customer receives products in a shorter time, especially in customized mass production environments.

In this work, there were four main axes of research in order to design the architecture. Regarding the manufacturing description language, the proposal takes into account that anyone with knowledge of how a product is manufactured, be able to describe, it with different levels of detail, how a product is completed.

This language provides a solution for the structured recording of the activities necessary to manufacture a product. Unlike traditional activity recording mechanisms that tend to be fuzzy and performed in natural language. Our language is structured with a general and universal syntax, without ambiguous redactions written with characteristics inherent to whoever writes them. Also, by using micro-motion theory in defining basic actions, industrial engineering experts can use it to conduct work studies. According to the experiments carried out in the paper, it was determined

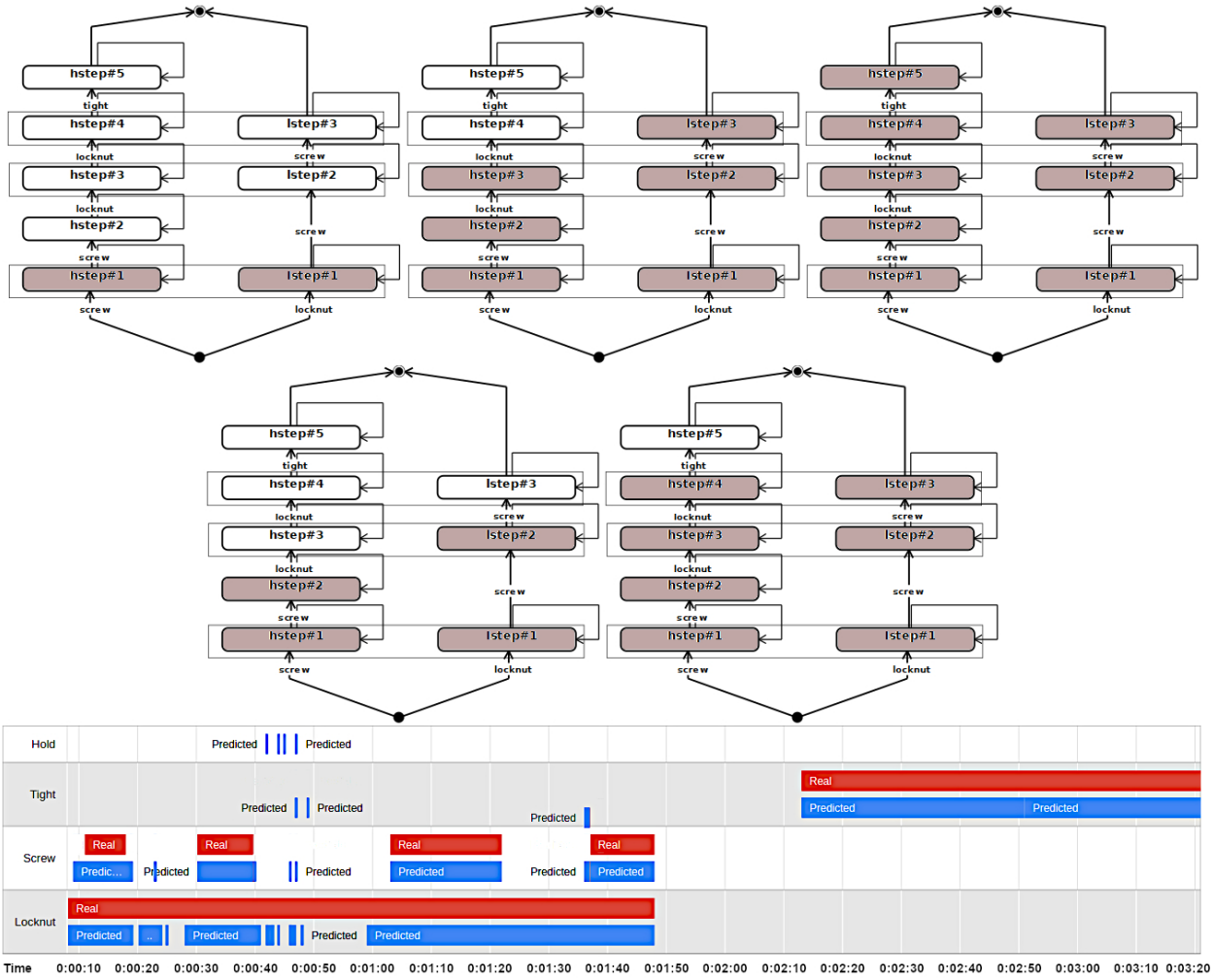


Figure 12: Execution of action graph with timeline of actions recognition

that the definition level is adequate for an operator to follow the instructions. Furthermore, the same language serves as a meta-language for describing the actions that an operator has to carry out to compare it with our visual control architecture. The experiments carried out in the present work will provide results for 95 % sensitivity.

The second axis of research was the development of a dataset built specifically for training the detection of tools and components using YOLO. For this purpose, a level of confidence in detection greater than 92% was achieved. These levels, being so high, were achieved due to the recovery of images of real tools and components, with augmented data for real images. In addition, synthetic images of the tools and components in different positions were added. This dataset is for general use and can be used by anyone who requires it.

Finally, the third axis of research is the action detection module as one of the fundamental points of this work since it detects, in real-time and with high levels of accuracy, what actions the operator is performing. The results obtained from this detection module motivate us to continue with new areas of study. Compared to manual labeling, the module has very high accuracy at the beginning and end of the task, which could generate "Task time" studies to delve into the areas of job design and manufacturing process engineering.

Regarding future lines of research, an action recognition module focused on the movements of the hands as the main element could increase even the performance since in the present paper the ADV was applied to the processing of the whole scene (including the static elements present that do not generate added value to the detection action recognition). Moreover, it is planned to design a list of valid actions for each tool and to validate against the tools associated with the identifier of objects that can only be performed in the language instructions sequence and the action recognition module. The advances of this project can also be used for the analysis of timings and movements in work design studies.

Acknowledgement

This work has been funded by the Spanish Government PID2019-104818RB-I00 grant for the MoDeaAS project, supported with Feder funds. This work has also been supported by University of Alicante grant for PhD studies UAFPU2019-13. We would also like to thank Nvidia for their generous hardware donation that made these experiments possible.

References

Abdelhameed, W. (2019). Industrial Revolution Effect on the Architectural Design. *2019 International Conference on Fourth Industrial Revolution, ICFIR 2019*, pages 1–6.

Abellan-Abenza, J., Garcia-Garcia, A., Oprea, S., Ivorra-Piqueres, D., and Garcia-Rodriguez, J. (2019). Classifying Behaviours in Videos with Recurrent Neural Networks. *Deep Learning and Neural Networks*, pages 965–980.

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning*. Springer USA.

Azorin-Lopez, J., Saval-Calvo, M., Fuster-Guillo, A., and Garcia-Rodriguez, J. (2016). A novel prediction method for early recognition of global human behaviour in image sequences. *Neural Processing Letters*, 43(2):363–387.

Borja-Borja, L. F. (2020). Arquitectura de visión y aprendizaje para el reconocimiento de actividades de grupos usando descriptores de movimiento. *Doctoral Thesis, University of Alicante*.

Cao, C.-Y., Zheng, J.-C., Huang, Y.-Q., Liu, J., and Yang, C.-F. (2019). Investigation of a Promoted You Only Look Once Algorithm and Its Application in Traffic Flow Monitoring. *Applied Sciences*, 9(17):3619.

Ericson, Gary; Franks, Larry; Rohrer, B. (2016). How to choose algorithms for Microsoft Azure Machine Learning.

ESRI (2020). How single-shot detector (SSD) works? Technical report, ESRI.

Everingham, M., Eslami, S. M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2014). The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, 111(1):98–136.

Fast-Berglund, Å., Fässberg, T., Hellman, F., Davidsson, A., and Stahre, J. (2013). Relations between complexity, quality and cognitive automation in mixed-model assembly. *Journal of Manufacturing Systems*, 32(3):449–455.

Ferreiro, S. and Sierra, B. (2012). Comparison of machine learning algorithms for optimization and improvement of process quality in conventional metallic materials. *International Journal of Advanced Manufacturing Technology*, 60(1-4):237–249.

Gong, M. and Shu, Y. (2020). Real-Time Detection and Motion Recognition of Human Moving Objects Based on Deep Learning and Multi-Scale Feature Fusion in Video. *IEEE Access*, 8:25811–25822.

Hedelind, M. and Jackson, M. (2011). How to improve the use of industrial robots in lean manufacturing systems. *Journal of Manufacturing Technology Management*, 22(7):891–905.

Hornung, A., Bennewitz, M., and Strasdat, H. (2010). Efficient vision-based navigation. *Autonomous Robots*, 29(2):137–149.

Khandelwal, R. (2019). SSD : Single Shot Detector for object detection using MultiBox.

Kong, Y. and Fu, Y. (2018). Human action recognition and prediction: A survey.

Krishna, R., Hata, K., Ren, F., Fei-Fei, L., and Niebles, J. C. (2017). Dense-Captioning Events in Videos. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob:706–715.

Lee, J., Bagheri, B., and Jin, C. (2016). Introduction to cyber manufacturing. *Manufacturing Letters*, 8:11–15.

Li, J., Gu, J., Huang, Z., and Wen, J. (2019). Application research of improved YOLO V3 algorithm in PCB electronic component detection. *Applied Sciences (Switzerland)*, 9(18).

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8693 LNCS(PART 5):740–755.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., and Berg, A. C. (2016). SSD: Single shot multibox detector. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9905 LNCS:21–37.

Luo, R. C., Hsu, Y. T., Wen, Y. C., and Ye, H. J. (2019). Visual image caption generation for service robotics and industrial applications. *Proceedings - 2019 IEEE International Conference on Industrial Cyber Physical Systems, ICPS 2019*, pages 827–832.

Lv, X., Dai, C., Chen, L., Lang, Y., Tang, R., Huang, Q., and He, J. (2020). A robust real-time detecting and tracking framework for multiple kinds of unmarked object. *Sensors (Switzerland)*, 20(1).

Makris, S., Karagiannis, P., Koukas, S., and Matthaiakis, A. S. (2016). Augmented reality system for operator support in human???robot collaborative assembly. *CIRP Annals - Manufacturing Technology*, 65(1):61–64.

Mancini, M., Karaoguz, H., Ricci, E., Jensfelt, P., and Caputo, B. (2018). Kitting in the Wild through Online Domain Adaptation. *IEEE International Conference on Intelligent Robots and Systems*, pages 1103–1109.

Martinez-Gonzalez, P., Oprea, S., Garcia-Garcia, A., Jover-Alvarez, A., Orts-Escolano, S., and Garcia-Rodriguez, J. (2020). Unrealrox: an extremely photorealistic virtual reality environment for robotics simulations and synthetic data generation. *Virtual Real.*, 24(2):271–288.

Melzner, J. and Bargstadt, H. J. (2016). A framework for 3D-model based job hazard analysis. *Proceedings - Winter Simulation Conference*, 2016-Febru:3184–3185.

Nackman, L. R., Lavin, M. A., Highsmith Taylor, R., Dietrich, W. C., and Grossman, D. D. (1986). AML/X: a programming language for design and manufacturing. *ACM '86: Proceedings of 1986 ACM Fall joint computer conference*.

Ni, B., Gang Wang, and Moulin, P. (2011a). Rgbd-hudaact: A color-depth video database for human daily activity recognition. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1147–1153.

Ni, B., Gang Wang, and Moulin, P. (2011b). Rgbd-hudaact: A color-depth video database for human daily activity recognition. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1147–1153.

Okubo, N., Nara, K., Takemura, S., and Ueda, Y. (2016). Applying an instructional design process to development of an independent verification and validation training program. *Proceedings - 2016 IEEE 29th Conference on Software Engineering Education and Training, CSEEandT 2016*, pages 237–240.

Park, S. and Kim, D. (2019). Study on 3D action recognition based on deep neural network. *ICEIC 2019 - International Conference on Electronics, Information, and Communication*, pages 5–7.

Patten, T., Vincze, M., Kropatsch, W., Wien, T. U., Fraundorfer, F., Roth, P. M., Schenk, F., Thalhammer, S., and Park, K. (2019). SyDD: Synthetic Depth Data Randomization for Object Detection using Domain-Relevant Background. *24th Computer Vision Winter Workshop*, pages 14–22.

Real, F., Batou, A., Ritto, T., and Desceliers, C. (2019). Stochastic modeling for hysteretic bit–rock interaction of a drill string under torsional vibrations. *Journal of Vibration and Control*, 1(X):107754631982824.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:779–788.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252.

Santoro, M., Marino, D., and Tamburrini, G. (2008). Learning robots interacting with humans: From epistemic risk to responsibility. *AI and Society*, 22(3):301–314.

Soomro, K., Zamir, A. R., and Shah, M. (2012a). UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402.

Soomro, K., Zamir, A. R., and Shah, M. (2012b). Ucf101: A dataset of 101 human actions classes from videos in the wild.

Tang, H., Peng, A., Zhang, D., Liu, T., and Ouyang, J. (2020). SSD real-time illegal parking detection based on contextual information transmission. *Computers, Materials and Continua*, 62(1):293–307.

Wang, L., Schmidt, B., and Nee, A. Y. C. (2013). Vision-guided active collision avoidance for human-robot collaborations. *Manufacturing Letters*, 1(1):5–8.

Wang, X., Chen, W., Wu, J., Wang, Y. F., and Wang, W. Y. (2018). Video Captioning via Hierarchical Reinforcement Learning. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4213–4222.

Xia, L., Chen, C., and Aggarwal, J. K. (2012). View invariant human action recognition using histograms of 3d joints. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 20–27.

Xiao, S., Wang, Z., and Folkesson, J. (2015). Unsupervised robot learning to predict person motion. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 691–696.

Xu, J., Wang, W., Wang, H., and Guo, J. (2020). Multi-model ensemble with rich spatial information for object detection. *Pattern Recognition*, 99:107098.

Yan, J., Yan, S., Zhao, L., Wang, Z., and Liang, Y. (2019). Research on Human-Machine Task Collaboration Based on Action Recognition. *Proceedings - 2019 IEEE International Conference on Smart Manufacturing, Industrial and Logistics Engineering, SMILE 2019*, pages 117–121.

Yang, Y., Cai, Z., Yu, Y., Wu, T., and Lin, L. (2019). Human action recognition based on skeleton and convolutional neural network. In *2019 Photonics Electromagnetics Research Symposium - Fall (PIERS - Fall)*, pages 1109–1112.

Yang, Y., Li, Y., Fermüller, C., and Aloimonos, Y. (2015). Robot learning manipulation action plans by "watching" unconstrained videos from the World Wide Web. *Proceedings of the National Conference on Artificial Intelligence*, 5:3686–3692.

Zhou, L., Cao, S., Liu, J., Tan, T., Du, F., Fang, Y., and Zhang, L. (2018). Design, manufacturing and recycling in product lifecycle: New challenges and trends. *4th IEEE International Conference on Universal Village 2018, UV 2018*, 2019-Janua:1–6.