Development of an AI-based digital twin for in-orbit AIT processes

Entwicklung eines KI basierten digitalen Zwillings für die In-Orbit AIT- Prozesse

Masterarbeit
Marcel Rohrmann | 2592972
Fachgebiet Datenverarbeitung in der Konstruktion

TECHNISCHE UNIVERSITÄT DARMSTADT

Development of an AI-based digital twin for in-orbit AIT processes

Entwicklung eines KI basierten digitalen Zwillings für die In-Orbit AIT- Prozesse

Masterarbeit
Marcel Rohrmann | 2592972
Fachgebiet Datenverarbeitung in der Konstruktion

Marcel Rohrmann
Matrikelnr.: 2592972
Studiengang: M.Sc. Maschinenbau

Masterarbeit
Thema: Development of an AI-based digital twin for in-orbit AIT processes
Entwicklung eines KI basierten digitalen Zwillings für die In-Orbit AIT Prozesse

Eingereicht: 02.06.2020

Betreuer: Vladimir Kutscher, M. Sc.
Thiago Weber Martins, M. Sc.
Martin Wende, M. Sc.

Prof. Dr.-Ing. Reiner Anderl
Fachgebiet Datenverarbeitung in der Konstruktion
Fachbereich Maschinenbau
Technische Universität Darmstadt
Hochschulstraße 1
64289 Darmstadt

Aufgabenstellung

Masterthesis

für

Marcel Rohrmann

TECHNISCHE UNIVERSITÄT DARMSTADT

Entwicklung eines KI basierten digitalen Zwillings für die In-Orbit AIT-Prozesse

Development of an AI-based digital twin for in-orbit AIT processes

Im Rahmen des Projekts „*Space Factory 4.0*“ wurde die Grundlage zur robotischen Montage von hochmodularen Satelliten auf einer In-Orbit-Plattform basierend auf Industrie 4.0-Prozessen erarbeitet. Dabei nimmt u.a. der Ansatz des digitalen Zwillings eine wichtige Funktion ein. Die Herausforderung liegt in der Verknüpfung von Daten, beispielsweise der Telemetriedaten aus dem Montageprozess sowie den Betriebsphasen von Kleinsatelliten mit dem entsprechenden virtuellen Modell. Diese großen Mengen und die Vielfalt von Daten bieten ein hohes Potenzial, Methoden der künstlichen Intelligenz (KI) einzusetzen, um Wissen aus den In-Orbit Prozessen zu gewinnen. Das bisherige Konzept des digitalen Zwillings von „Space Factory 4.0“ enthält keine KI-Methoden.

Daraus leitet sich das Ziel dieser Masterarbeit ab, den Einsatz von KI-Methoden im Rahmen von In-Orbit Assembly, Integration and Testing (AIT) Prozessen zu untersuchen. Es soll ein Konzept entwickelt werden, welches die bisherigen Ergebnisse von „Space Factory 4.0“ mit KI-Methoden aufwertet. Mithilfe des entwickelten Konzepts soll der Nutzen der KI-Methoden in Zusammenhang mit digitalem Zwilling und den In-Orbit AIT Prozessen quantifiziert werden.
Das Konzept soll prototypisch implementiert werden, um dessen Funktionsfähigkeit demonstrieren zu können.

Arbeitspakete:

- Recherche geeigneter KI-Methoden im Umfeld von Industrie 4.0 und AIT-Prozessen der Raumfahrt
- Erarbeitung von Anwendungsfällen von KI-Methoden im Rahmen von Space Factory 4.0
- Ermittlung der Anforderungen aus ausgewählten Anwendungsfällen
- Konzept eines KI-basierten digitalen Zwillings mit ausgewählten KI-Methoden und Anwendungsfällen
- Implementierung und Validierung anhand eines Demonstrators
- Dokumentation und Präsentation der Ergebnisse

Tag der Ausgabe: 01.12.2019
Betreuer: Vladimir Kutscher, M.Sc., Thiago Weber Martins, M.Sc., Martin Wende, M.Sc.

Prof. Dr.-Ing. R. Anderl

Fachgebiet Datenverarbeitung in der Konstruktion

Department of Computer Integrated Design

Prof. Dr.-Ing. Reiner Anderl

Otto-Berndt-Straße 2
64287 Darmstadt

Tel. +49 6151 16 - 21791
Fax +49 6151 16 - 21793
anderl@dik.tu-darmstadt.de

Datum
25.11.2019

Seite: 1/1

Abstract

The Space Factory 4.0 was created to investigate the potential use of Industry 4.0 technologies for manufacturing in space. Due to rapid increase of new achievements in artificial intelligence, this thesis focuses on how artificial intelligence technologies can be integrated in the existing Space Factory 4.0 process. Therefore, three major AI technologies in the field of Industry 4.0 are introduced: Predictive Maintenance, Self-learning Production System and Computer Vision. The fundamentals of all three technologies are explained and the state-of-the-art is presented. In the end three concepts for each AI technology are developed.

The use of Predictive Maintenance is investigated in more detail. For this purpose, the sensor data of a servo motor is analyzed. The servo motor is integrated in a robotic arm, which is used for the assembly process of satellite parts. The servo is visualized and a machine learning algorithm is applied. This supervised learning algorithm is called Support Vector Machine. It is trained with amperage data to classify if force is applied to the servo or not.

A Computer Vision system was also implemented in the Space Factory 4.0. Therefore, a webcam was installed to the robotic arm. With the integration of a Convolutional Neural Network the system is able to detect multiple objects in real-time. Built upon, machine vision applications are applied to detect smaller details of the detected object. For demonstration a physical CubeSat module is 3D printed and a digital twin created. The Computer Vision system is able to measure dimension and surface color of the physical CubeSat module and to synchronize it with the corresponding digital twin.

Declaration

Hiermit versichere ich, Marcel Rohrmann, die vorliegende Master-Thesis / Bachelor-Thesis gemäß § 22 Abs. 7 APB der TU Darmstadt ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß § 23 Abs. 7 APB überein.

Datum / Date:	Unterschrift/Signature:
19.05.2020	

Table of contents

List of figures

List of tables

Abbreviations

AI	Artificial Intelligence
AIT	Assembly, Integration and Testing
API	Application Programming Interface
CAD	Computer-aided Design
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CPS	Cyber-physical System
CPU	Central Processing Unit
CSV	Comma-seperated Value
DiK	Datenverarbeitung in der Konstruktion
DOF	Degree of Freedom
ECSS	European Cooperation for Space Standardization
ERP	Enterprise Resource Planning
ESA	European Space Agency
FOV	Field of View
GPU	Graphics Processing Unit
IoT	Internet of Things
mAP	mean Average Precision
MES	Manufacturing Excecution System
PLM	Product Lifecycle Managment
RAMI 4.0	Referenzarchitekturmodell für Industrie 4.0
RUL	Remaining Useful Life
SVC	Support Vector Classifier
VR	Virtual Reality
XML	Extensible Markup Language

1 Introduction

The number of launched satellites is rapidly increasing since several years. Mega-constellations of 12000 satellites like Starlink from SpaceX underline this as an important example [1]. Furthermore, as Jeff Bezos is quoted “The reason we go to space, in my view, is to save the Earth. [If] we are going to continue to grow this civilization, …, we need to move heavy industry off Earth. It will be better done in space anyway.” [2]. The traditional method of Earth-build-and-launched satellites is associated with high costs, which is not profitable for every satellite manufacturer. To face this challenge, the institute “Datenverarbeitung in der Konstruktion” (DIK) at TU Darmstadt researches the potential of manufacturing in space. This project is called Space Factory 4.0 [3, 4]. The idea is to apply new technologies from Industry 4.0 and research potential benefits when manufacturing small satellites directly in space. Since space agencies all around the world are investigating the potential use of asteroid mining, the Space Factory can also be a pivotal element to use resources from other planets or asteroids. But how to control an autonomous factory over 400 km up in space?

This thesis researches how new achievements in artificial intelligence can be applied to the Space Factory 4.0 process. One major challenge is the communication with the Space Factory 4.0. The factory is orbiting the earth like a typical satellite and a permanent data link is not always possible as visualized in Figure 1. The ground station has a limited angle of view (red lines) and the telecommunication can only occur when the space factory passes through this area. Therefore, one key element of the Space Factory 4.0 is the Centralized Information Base (CIB). It shall serve as the main database of the whole process. All data, from space or earth, is stored in this centralized database and data can be shared with all applications in the network. This thesis focuses on how this data can be further analyzed with artificial intelligence. Primarily it will be researched how neural networks and deep learning can be implemented to increase the autonomy of the Space Factory 4.0. Convolutional Neural Networks will be deployed to detect objects so that a robot can understand its surrounding.

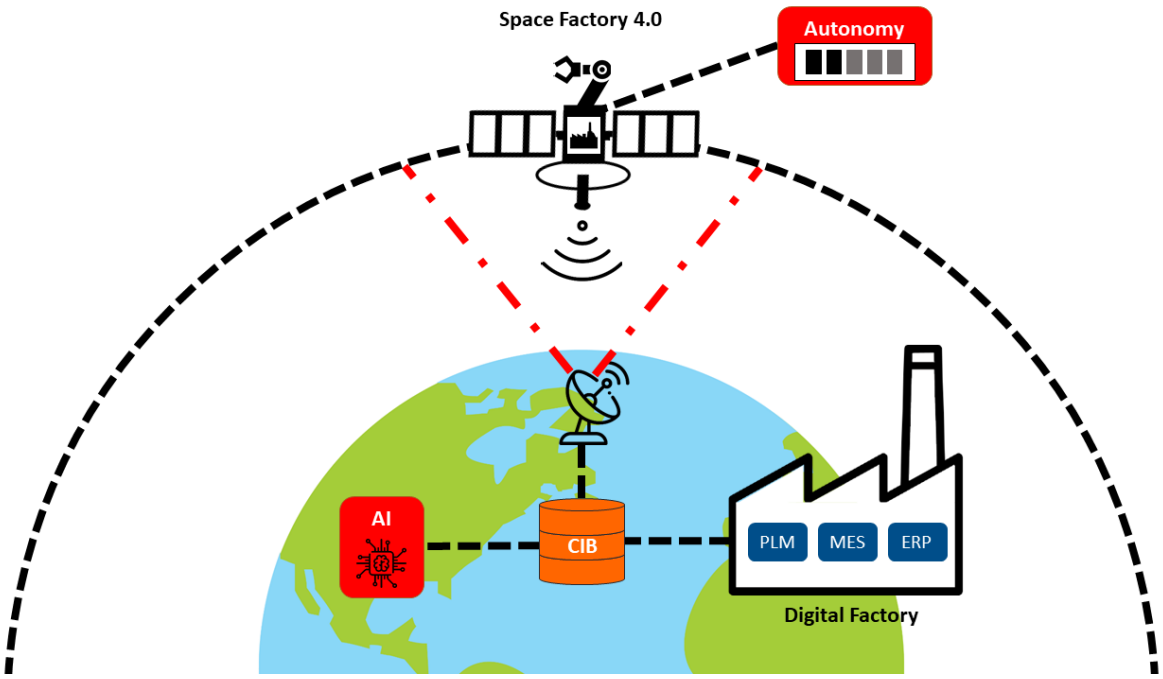


Figure 1: Space Factory 4.0 concept enhanced with artificial intelligence.

The benefits of machine learning algorithms will be analyzed. Therefore, the data of machines is collected to predict anomalies or health condition. Even new achievements in reinforcement learning will be introduced. The goal is that with reinforcement learning, the Space Factory 4.0 will be able to learn operations by itself only by getting rewards from the algorithm and environment. In the end, the Space Factory 4.0 would become a full Self-Learning Production System with operations more precise than if they were developed by a human.

Following this goal, this work focuses on three major technologies: Predictive Maintenance, Self-Learning Production System and Computer Vision. For understanding these technologies, the fundamentals of artificial intelligence, Industry 4.0 and space systems will be clarified in chapter 2. Chapter 3 contains the main features of machine learning. For Predictive Maintenance, state-of-the-art supervised learning algorithms are described in 4.1 and the implementation in the space factory process and a small pilot program presented in chapter 5.5. The theory of the Self-Learning Production System, heavily based on reinforcement learning, is described in chapter 4.2 and the first concept and requirements are introduced in chapter 5.6. Hence, Computer Vision is a pathbreaker and critical technology for this networked production process. It is deeply researched in chapter 4.3. No other component already collects and interprets as much data as image processing. It is necessary to verify and process what has been seen in every phase of production and to transmit the results to the systems in the network. Especially to achieve high automation, Computer Vision is necessary for controlling the robots in the Space Factory 4.0. It alone is capable of supplying extensive and important secondary information to all systems operating in the process network. Only the extensive collection and evaluation of this vision data makes reliable, intelligent and autonomous action possible. A detailed concept of the integration of object detection and machine vision is implemented in chapter 5.7 with an evaluation of the system in chapter 6. This is followed by an outlook and a summary of the thesis.

2 Fundamentals

In this chapter, the fundamentals of artificial intelligence are described. It is intended to provide an overview of the far reaches of artificial intelligence and how it is linked to space exploration. Building upon the principles of Industry 4.0 elements are explained, such as digital twin and Internet of Things. The core concept of the Reference Architecture Model for Industry 4.0 (RAMI 4.0) is presented and how artificial intelligence can be integrated into this process.

2.1 Definition

To date, there is no generally accepted, clear and precise definition of the term “Artificial Intelligence” (AI). However, there are many different approaches to define artificial intelligence. From a general point of view, artificial intelligence is a branch of computer science that deals with methods and technologies that enable a computer to solve tasks that, when solved by humans, require intelligence [5]. AI refers to a broad field of science encompassing not only computer science but also psychology, philosophy, linguistics and other disciplines. Other definitions are:
“AI is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable.” By Prof. John McCarthy [6].
“Artificial intelligence is a computerised system that exhibits behaviour that is commonly thought of as requiring intelligence.” By the National Science and Technology Council [7].
“Artificial intelligence is the attempt to pass a sophisticated version of the Turing Test” at the 8th IJCAI held in Karlsruhe [8].

2.2 Branches of artificial intelligence

An AI system combines and utilizes mainly machine learning and other types of data analytics methods to achieve artificial intelligence capabilities. The field of AI has expanded to various fields, as shown in Figure 2.

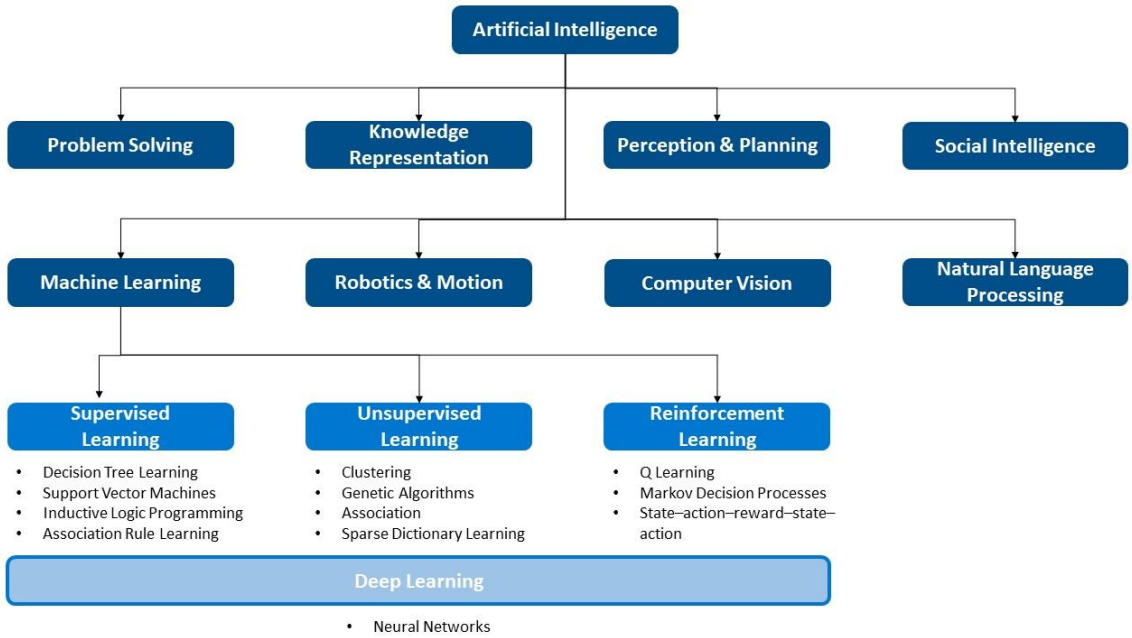


Figure 2: Overview of sub branches of artificial intelligence (based on [9])

The terms machine learning, robotics and smart machines are often used in relationship to AI, or sometimes even as synonyms. Machine learning can be further subdivided into supervised, unsupervised and reinforcement learning. An extract of their correlating algorithms are shown in Figure 2. AI is a complex field of interest, with a lot of different branches. The branches relevant for the Space Factory 4.0 project will be described in more detail in the following chapters.
The principles of machine learning are explained in chapter 3 with a particular focus on supervised learning algorithm in chapter 4.1. Reinforcement learning is further explained in chapter 4.2.
This thesis focuses mainly on Computer Vision and the use of neural networks, which will be elaborated in chapter 4.3 continuously.

2.3 Evolution of artificial intelligence

Alan Turing, a Cambridge mathematician of the first half of the twentieth century, can be considered as one of the first researchers of artificial intelligence in computer science. In 1950 Turing published the very first paper suggesting the possibility of artificial intelligence [10]. It is called the Turing Test. Research on artificial intelligence came to an end with the publication of the Minsky and Papert book “Perceptrons” (1969) in which they showed that the perceptron is incapable of learning to classify as true or false the inputs to such simple systems as the exclusive “or” (XOR – either A or B but not both). Though this statement is false, the research on artificial intelligence stopped, which is also referred to as “AI Winter” [10]. Research started again when the work of D.E. Rumelhart revived immense interest in neural networks, fuzzy sets, statistical and genetic algorithms [11]. In the 1990s, distributed AI got increasingly focused along with the emergence of multi-agents, which made previous AI systems in centralized, hierarchical control structures flat [12]. In the 2000s, Web 2.0, web services and web intelligence emerged in AI systems. The use of the Internet benefited that large volumes of data are available online. This interest for AI continued and around 2010, was promoted by three factors associated with each other: the sources of Big Data, improved machine learning algorithms and more powerful computers. [13]

Table 1: Evolution of artificial intelligence [12]

Age	**1950s-1960s**	**1980s**	**1990s**	**2000s**	**2010s**
Computation	Mainframes	PC	PC	Networks	Things + Cloud
Content/ Focus	DBMS structured content; Knowledge representation	Soft computing; Data analytic	Distributed computing intelligence	Unstructured user-created content; Web intelligence	IoT; Big Data; Machine Learning; Deep Learning
Control Structure	Centralized	Centralized	Distributed	Web-service based	Cyber-physical-system

2.4 Artificial intelligence in space

As described in the previous chapter, artificial intelligence consists of several branches. Hence, there exists a wide range of AI applications in the space industry. One major branch in space is

the use of robotics. Already several rovers have been sent to the Moon or Mars. Especially on Mars, the long time delay of the radio signal (on average 12 min) is a challenge [14]. Therefore, several functions have to be automated with the help of artificial intelligence. For Example, the Mars 2020 rover mission is prototyping the use of onboard scheduling software. The current plan is to have an onboard scheduler help the rover independently adapt to changes on the surface of Mars, enabling the rover to operate more efficiently. Specifically, if activities finish earlier or later than expected, the onboard scheduler will allow the rover to extend or add activities to perform, e.g. science experiments later [15]. A similar project is the Autonomous Sciencecraft Experiment operating onboard the Earth Observing-1 satellite. Whose software uses onboard continuous planning, robust task and goal-based execution, onboard machine learning and pattern recognition [16].

More artificial intelligence in space projects can be found by the ESA Advanced Concept Team or the NASA JPL Artificial Intelligence Group [17, 18]. ESA is researching artificial intelligence applications for satellite operations, in particular to support the operational side of large satellite constellations, which includes relative positioning, communication, end-of-life management [19].

Another project focusing on deep learning is CIMON (Crew Interactive MObile companioN). The device is an AI-based assistant for astronauts developed by Airbus and IBM and funded from the German Aerospace Center (DLR) [20]. An artificial neural network has been tested for training voice signal and for detecting anomaly signal on multiple analog sensors [21]. CIMON can orientate itself using a stereo camera and a high-resolution camera that it uses for facial recognition. Two other cameras at the sides are used for photo and video documentation. With the ultrasound sensors, it can measure distances, recognize potential collisions and navigate autonomous trough the International Space Station [22].

Artificial intelligence also contributes to spacecraft guidance dynamics and control [23]. An example is the research of C. Sanchez-Sanchez. It studies how Deep Neural Networks can be trained on optimal state-feedback of continuous-time, deterministic, non-linear systems like inverted pendulum stabilization and pinpoint landing of a spacecraft [24]. Another work is concentrating on Reinforcement Learning to design a rendezvous-mission, where one satellite needs to meet a second satellite within a constellation while avoiding collisions. The developed algorithm allows the satellite controller to make sequential decisions on the path to take, safely navigating the environment [25].

Even so, there are many projects mentioned with AI. According to ESA, implementing artificial intelligence in space missions is starting slowly. New capabilities are adopted only when they offer overwhelming benefits to a mission. Even then, a new capability's risks must be well understood and aggressively retired. The damage an artificial intelligence system can make in space is several magnitudes higher than in an industrial environment on earth, because one failure in space most of the time cannot be fixed [19].

2.5 Space Factory 4.0

Space Factory 4.0 is a joint project between the TU Darmstadt, TU München and the Zentrum für Telematik (ZtF). The project is funded by the DLR and the Bundesministerium für Wirtschaft und Energie (BMWi) with further subcontractors being OHB Systems, Von Hoerner & Sulger (VH&S) and Telespazio VEGA Germany.

Space Factory 4.0 has been founded to establish new processes and technologies for rapid satellite assembly on an in-orbit platform with Industry 4.0 processes. The satellites are highly modular for a robotic assembly. One key feature is the seamless use of digital twins in the overall process [26]. For teleoperation of the assembly robot, a Human-Machine-Interface has been developed.

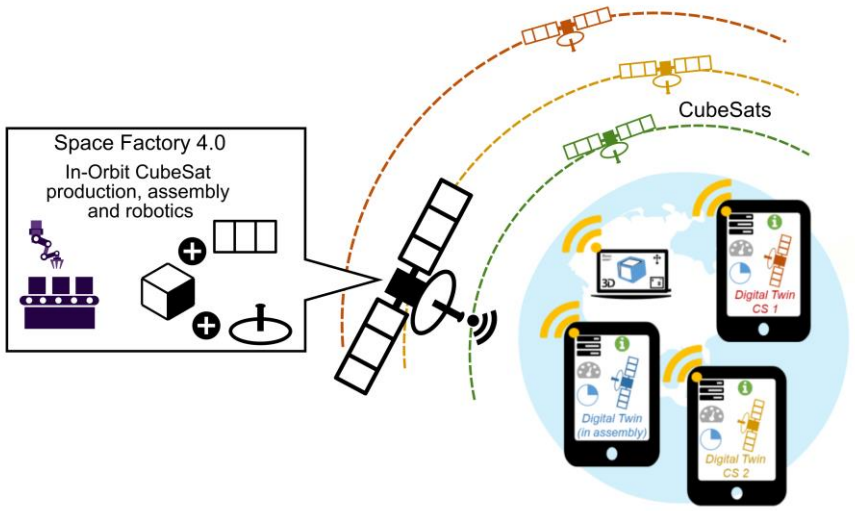


Figure 3: Space Factory 4.0 concept [26]

The department “Datenverarbeitung in der Konstruktion” (DiK) at TU Darmstadt mainly concentrates on the research of integrating digital twin and Industry 4.0 approaches in the overall process. Therefore, a small test setup of the Space Factory 4.0 has been developed, which will be described in chapter 5.3. Artificial intelligence has not yet been applied to the Space Factory 4.0 and the benefits will be researched in this thesis.

2.6 Industry 4.0

Industry 4.0 (also called the fourth industrial revolution) represents the consistent further development of three preceding industrial revolutions. It is based on historical and technological approaches of the intelligent factory and computer-integrated production. The core of industry 4.0 build the Cyber-Physical System (CPS) and the Internet of Things (IoT). It is called the fourth industrial revolution because of the previous industrial leaps shown in Figure 4. The first industrial revolution took place with the development of mechanical power machines using water and steam (steam engines). It enabled the first real industrialization in the textile, iron and steel industry. The first mechanized loom was developed in 1784. The second industrial revolution is the industrial mass production based on the division of labor with the help of assembly lines and conveyor belts with the help of electrical energy from 1870 onwards. Driven by the German economic miracle at the beginning of the sixties of the 20th century, the third industrial revolution that still prevails today. This is characterized by the use of electronics and information and communication technology to automate production. [27]

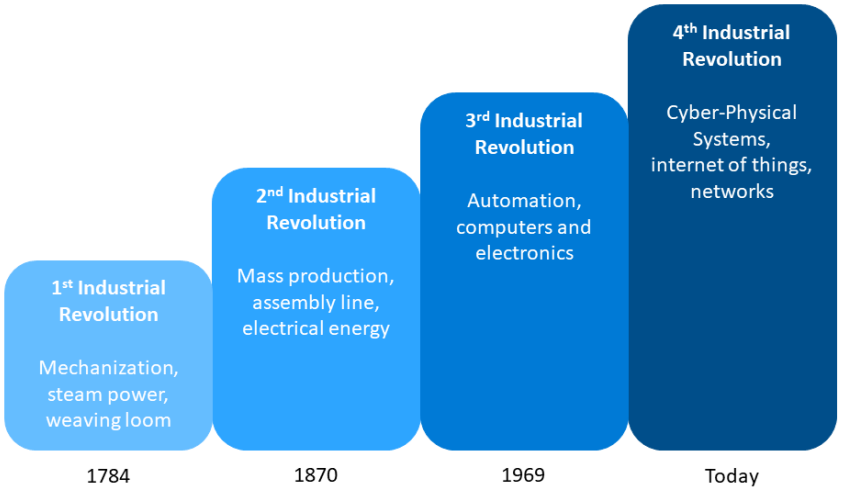


Figure 4: The evolution of the industry [28]

A summary of the differences between Industry 3.0 and 4.0 is shown in Table 2. In Industry 3.0, IT and computer systems are used to automate processes. These processes already operate primarily without human interaction, but the human aspect is still present. Where in Industry 4.0, the availability and use of vast quantities of data on the production floor is in the center of attention. Internet of things with Cyber-Physical System allows to gather and interpret data in ways that were not able before. [27]

Table 2: Change from Industry 3.0 to Industry 4.0 [27]

	Industry 3.0	**Industry 4.0**
Production control	Centralized	Decentralized (CPS)
Control technology	Monolith	Open internet standard (Cloud)
Data processing and provision	Time-delayed	Real-time
Right of use	License cost	Pay per use
Software management	Software suite	Cloud-Apps (SaaS)

2.6.1 Cyber-Physical System

Cyber-Physical Systems (CPS), can be explained as supportive technology for the organization and coordination of networking systems between its physical infrastructure and computational capabilities. In this respect, physical and digital tools should be integrated and connected with other devices in order to achieve decentralized actions. In other words, embedded systems generally integrate physical reality concerning innovative functionalities, including computing and communication infrastructure. [29, 30]

2.6.2 Digital twin

There exists no single valid definition for what exactly a digital twin is and different defintions will be introduced. In 2003 digital twin was defined by John Vickers as a virtual representation of a physical product or process [31]. With the help of an interface between the real and digital objects, all relevant information is exchanged. A digital twin contains not only pure data, but also algorithms that accurately describe its corresponding counterpart in the real world. “Digital twin” is a broad term since first defined by M.Grieves in 2002 [32]. Since 2010 the terminology was influenced heavily by NASA defining a digital twin as an “integrated multi-physics, multi-scale, probabilistic simulation of a vehicle or system that uses the best available physical models, sensor updates or fleet history to mirror the life of its flying twin. The digital twin is ultra-realistic and may consider one or more important and interdependent vehicle systems.” [33]. A digital twin is an I4.0 component defined in RAMI 4.0. It contains all information used in all phases of the product life cycle. The digital twin evolves and continuously updates to reflect any change to the physical counterpart throughout the product lifecycle. It is a crucial technology for the digitalization of the world [34]. At DiK a digital twin is defined as a “digital representation of a physical object, in particular manufactured parts, assemblies and products existing in reality” [35]. It should contain as many properties as possible to enable an object representation close to reality.

Besides digital twin, the term digital shadow is common. The digital shadow transfers the real production process into the virtual world. Moreover, according to T. Bauernhansl, “Digital shadow is the representation of the processes in production, development and adjacent areas with the purpose of creating a real-time capable evaluation basis of all relevant data" [36]. In detail, this includes the description of the necessary data formats, data selection and the level of granularity. The digital shadow is a preliminary stage of the digital twin. Building on the data of the digital shadow, the digital twin can provide an almost identical representation of reality through a process model and simulation. The data of the digital shadow can be used for further data analysis. The five steps are: Descriptive, Diagnostic, Predictive, Prescriptive analysis and are visualized in Figure 5. [36]

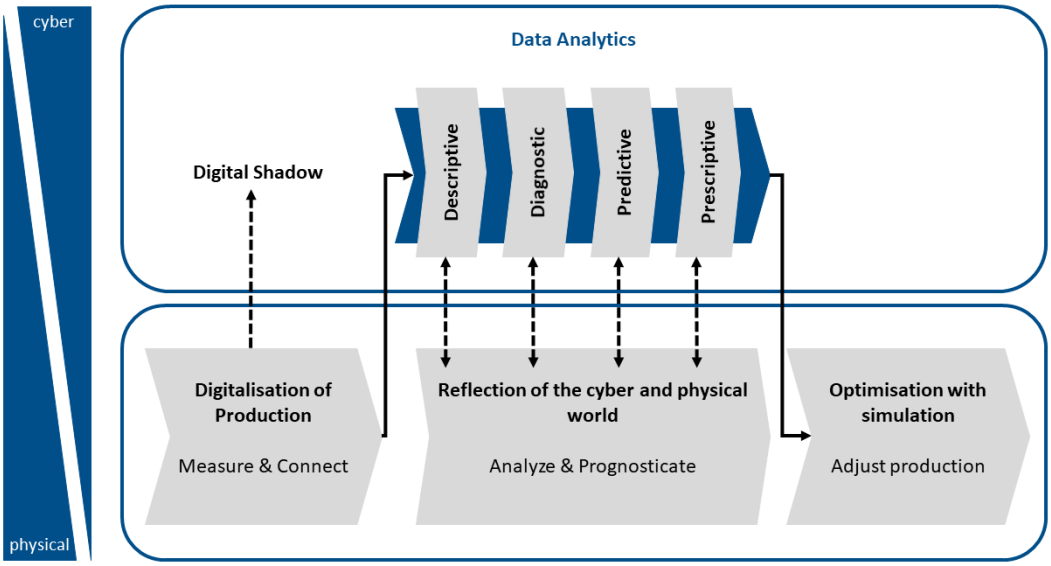


Figure 5: Stages of data evaluation and analysis (translated from [36])

According to U. Bracht trough out the year’s different terms and definitions have been developed to describe the digitalization process in a company. To summarize all these different approaches, the term digital factory is instituted. The digital factory is the generic term for a

comprehensive network of “digital models, methods and tools - including simulation and three-dimensional visualization - which are integrated by a continuous data management. Its goal is the holistic planning, evaluation and continuous improvement of all essential structures, processes and resources of the real factory in connection with the product" [37].

2.6.3 Internet of Things

Communication and networking is another asset in Industry 4.0. It can be described as a link between physical and distributed systems that are individually defined. Using communication tools and devices, machines can interact to achieve given targets. Internet of Things (IoT) relies on both smart objects and smart networks. It also enables a physical objects integration to the network in manufacturing and service processes. In other words, the primary aim of IoT is to provide computers and machines to see and sense the real world. IoT Applications can provide connectivity from any time, anywhere, for anyone for anything [38]. Thanks to the recent advances in decreasing costs for sensor networks, NFC, RFID and wireless technologies, communication and networking used for IoT suddenly became an engaging topic for the industry [29].

2.6.4 RAMI 4.0

The Reference Architecture Model for Industry 4.0 (RAMI 4.0) consists of the Industry 4.0 component and its administration shell. The working group I “Referenzarchitekturen, Standards und Normung“ aims to define the essential requirements of an Industry 4.0 system consisting of interacting components [39]. The basis is a fundamental interaction mechanism that is based on basic standards with interpretable semantics, allows plurality and interoperability of the communicating assets and allows interaction across system boundaries of different manufacturers and users. RAMI 4.0 is constructed from three dimensions, such as Layers, Life Cycle & Value Steam and Hierarchy Levels, as shown in Figure 6.

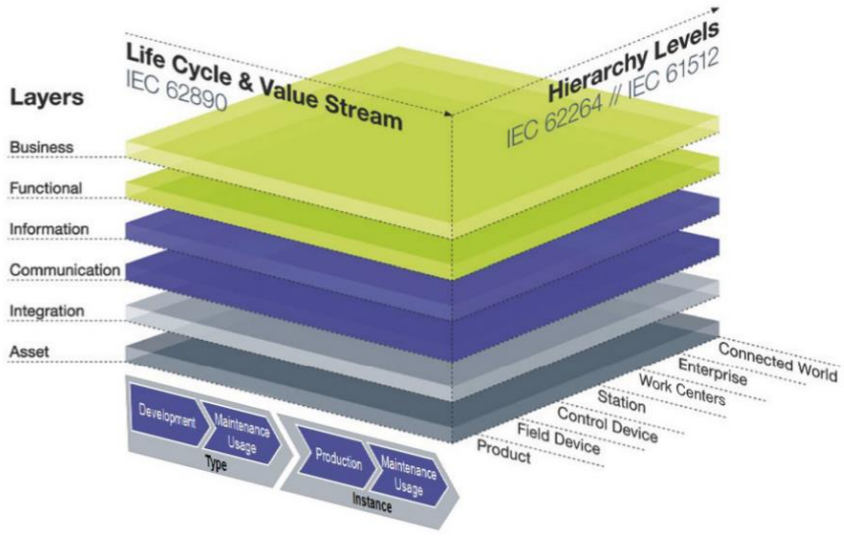


Figure 6: Reference Architecture Model for Industry 4.0 [39]

Layers are used in the vertical axis to represent the various perspectives, such as data maps, functional descriptions, communications behavior, hardware/ assets or business processes. This corresponds to IT thinking, where complex projects are split up into clusters of manageable parts. A further important criterion is the product life cycle with the value streams it contains. This is displayed along the left-hand horizontal axis. Dependencies, e. g. constant data

acquisition throughout the life cycle, can therefore also be represented well in the reference architecture model. The third important criterion, represented in the third (right-hand horizontal) axis, is the location of functionalities and responsibilities within the factories or plants. This represents a functional hierarchy, and not the equipment classes or hierarchical levels of the classical automation pyramid. The issue here is not implementation, but solely functional assignment. For classification within a factory, this axis of the reference architecture follows the IEC 62264 and IEC 61512 standards. [39, 40]

2.6.5 Integration of AI in RAMI 4.0

Key technologies such as AI, which depend on a lot of information from a wide variety of sources, require the following essential features, which are made available through Industry 4.0 and the administrative shell:

- Provision of the information about trustworthy components in the network
- Information flows must be structured and controllable
- The mechanism must allow a decentralized data source structure as well as centralized data storage
- Information, functions and their consequences must be clearly defined
- Collecting, evaluating and learning from information
- Active interaction between data sources, AI learning mechanisms and human decisions
- Conversion of a learning outcome into controlling functions of the administrative shell
- Conversion should also be possible in a changing system

AI technologies can make decisive contributions to the autonomy and responsiveness of a system. Also, the symbolic closing as an essential basis for automated decision making is of high importance. Artificial intelligence makes it possible to connect semantics automatically. Pattern recognition and the detection of anomalies is another focus in the use of AI. These capabilities support agile behavior. [5]

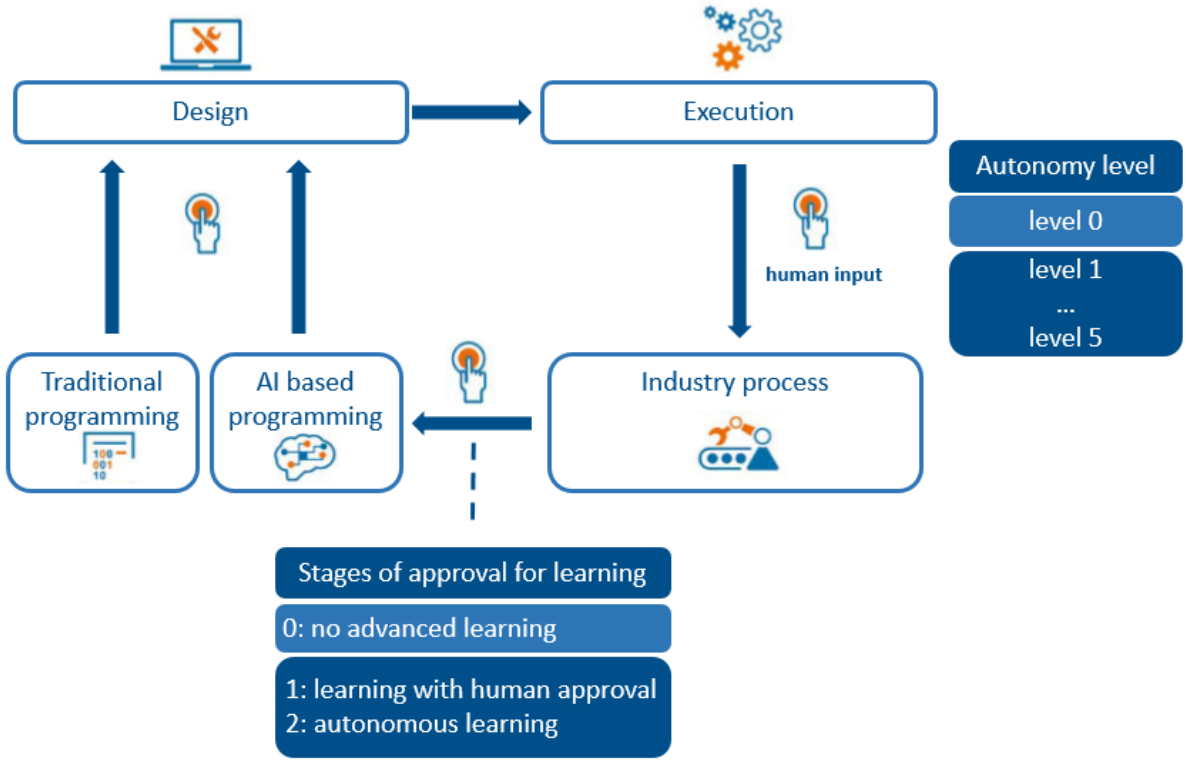


Figure 7: Control loop of traditional and AI-based programming [5]

For simple automation procedures in the industrial process control, traditional programming and additional human intervention are sufficient in most cases. Nevertheless, for complex processes with a higher decision-making volume, programming based on AI technologies is increasingly useful. In contrast to traditional programming, the use of AI does not always produce the same result because of the extensive data and information pools used for the learning process. Instead, the use of AI is aimed at constant process optimization and "correct" decision-making in problem cases. At the same time, it is permitted that the AI system proposes an unpredicted solution. However, the decision made should always be plausible and, if possible, correspond to an intended procedure. The influence of AI on industrial processes can be visualized in Figure 7. It can be divided into the phase of decision making (“Design”) and action execution (“Execution”). In an AI-supported design, the system is trained through initial learning and together with traditional programming developed to execute an industrial process. The AI influence that is decisive for the control part "Execution" can be classified according to the autonomy levels 0-5 described in Table 3. [5]
At level 0 the system does not have any AI-based automatisation and the human has full control and responsibility. Level 1-4 are called semi-autonomous systems. In this case AI programming was used, but human intervention is still required. A complete autonomous operation of production under AI control is reached at level 5. The AI program replaces any human intervention. [5]

Table 3: Levels of autonomous systems (according to [5, 41, 42])

Autonomy Level	Definition
Level 0	No autonomy; human has full control without assistance
Level 1	Operations Assistance; automation system provides decision support for necessary operations; humans always responsible
Level 2	Occasional autonomy in certain situations; humans pull support, e.g., for plant start-up; humans always responsible
Level 3	Limited autonomy in certain situations. System alerts to issues; humans confirm proposed solutions or act as fallback
Level 4	System in full control in certain situations; humans supervise actions
Level 5	Autonomous operation in all situations; humans may be completely absent.

In the "Design" phase of the industrial process the human being as a supervisor decides in advance which data will be accepted for the initial learning phase. The human also has to decide which data the AI system may receive for a further learning process during process control. On the one hand, data is necessary to optimize a process, but on the other hand, overfitting must be avoided. This is because the overfitting of data evaluation can also worsen an existing solution. This approval process for automatic learning can be roughly divided into three stages:

Table 4: Approval stages in the AI learning process [5]

Approval stage	Description
Stage 0	Learning manually; no approval for advanced learning; new learning is only possible by a manual extension (e.g., adding of new rules)
Stage 1	Event-driven learning with human approval; AI collects further training data and creates knowledge independently through continuous learning; new knowledge must be approved by humans
Stage 2	Autonomous learning; AI collects data independently, generates new knowledge, adapts its behavior within the system limits; decision making automatically, without human input, only within predefined limits

Still, there can be a tendency to confuse artificial intelligence with autonomous systems. While AI indeed has a close relation to autonomous systems, it is a technological means through which a specific level of autonomy can be achieved. Figure 8 depicts the relation between AI and autonomy. To achieve higher autonomy in a system, more intelligence is needed in the system. Such intelligence can be provided by specific AI technologies.

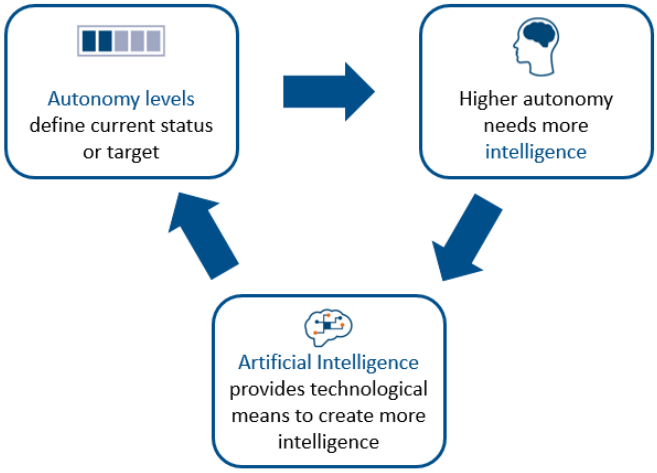


Figure 8: Relation between autonomy and artificial intelligence [5, 41]

2.7 Assembly, Integration and Testing

A spacecraft development can be divided into 5 phases [43]:

1. Conceptual design (phase A)
2. Preliminary design/analysis (phase B)
3. Detailed design analysis (phase C)
4. Development (fabrication, assembly, integration, testing) (phase D)
5. Operation (phase E)

The Assembly, Integration and Testing (AIT) is in 4th phase. It includes assembling the various mechanical, electrical, and thermal subsystems into an integrated spacecraft and performing tests on the integrated spacecraft to ensure that it will operate properly in the environment in which it must function. [43]

Integration is the process of assembling components into a subsystem and ensuring that the individual units perform properly when interconnected. Testing, or also named verification, is the process of determining that the part, component, subsystem or system meets specified requirements and will operate properly under the environmental conditions expected to be encountered during the mission. There are several testing procedures defined by ESA in the ECSS-E-ST-10-03C. Testing is part of the system engineering process and starts at the early phase of the mission when defining the verification process in terms of the model philosophy and test sequence and ends at the last testing phase prior to launch. Testing is categorized into the following categories: General (Functional, Performance etc.), Mechanical (Vibration, static load etc.), Structural integrity (Leak, Proof pressure etc.), Thermal (Thermal vacuum), Electrical/Radio frequency (EMC, Magnetic etc.), Mission specific (audible noise). [44], [43]

A more detailed test sequence of space equipment is shown in Figure 9.

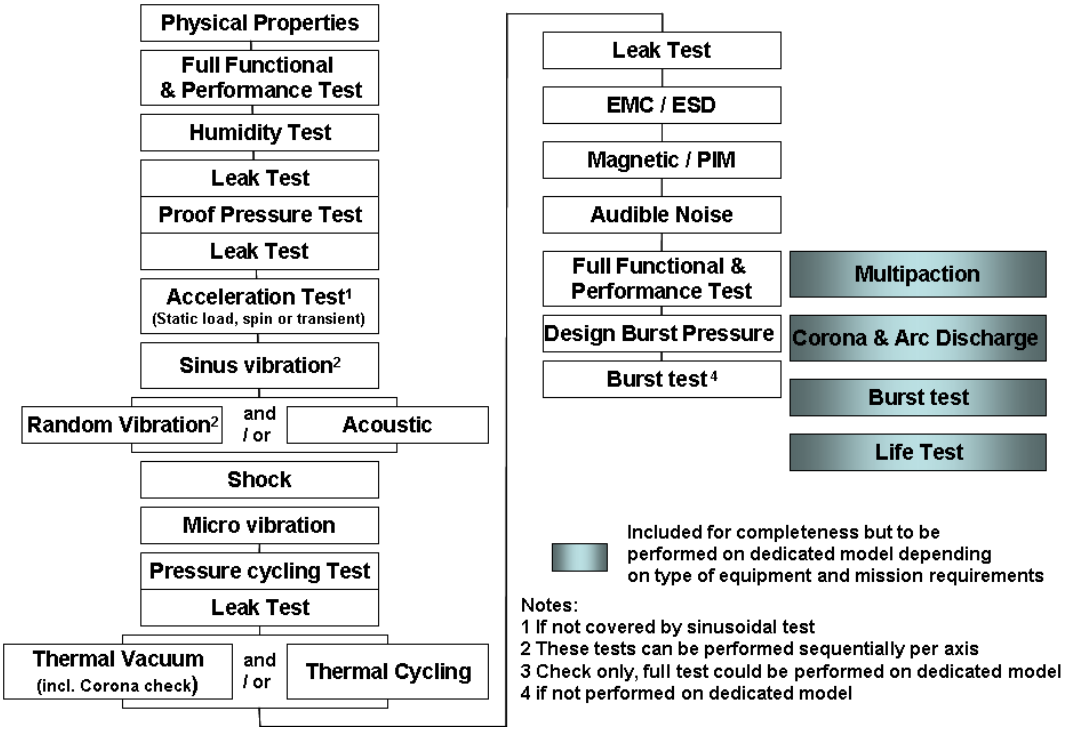


Figure 9: Space segment equipment test sequence according to ECSS [44]

According to ECSS, experience has demonstrated that incomplete or improper on ground testing approach significantly increase project risks leading to late discovery of design problems, workmanship problems or in-orbit failures.

2.8 CubeSat

The CubeSat standard was created by California Polytechnic State University, San Luis Obispo and Stanford University's Space Systems Development Lab in 1999 to facilitate access to space for university students. Since then, the standard has been adopted worldwide. Also the Space Factory 4.0 is designed for highly modular satellites, e.g, CubeSats. CubeSats, as mentioned in the name, have a cubic form. One unit of a CubeSat (referred to as 1U) has a size of 10x10x11.35 cm and maximum weight of 1.33 kg [45]. These standardised cubes can be stacked up to build bigger satellites, as shown in Figure 10. Sizes of up to 6U or 12U have already been developed.

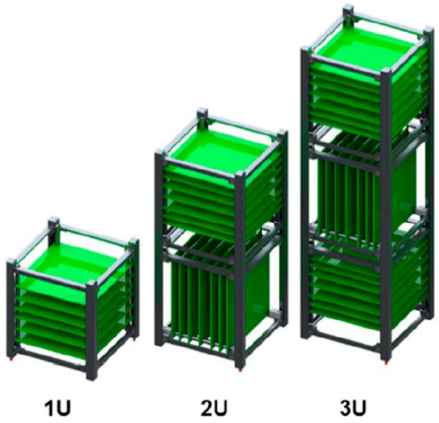


Figure 10: Different sizes of CubeSats [46]

Inside of the CubeSat structure are subsystems and components stacked up (green module). Some of these modules can be: front access board (FAB), backplane (BP), electrical power system (EPS), accurate attitude determination and control systems (ADCS), 3-axis reaction wheel for axes control, on-board computer (OBC), image acquisition component (IAC), communication system (COMM) and further mission-related components (e.g., scientific experiment). [47, 48]

2.9 Satellite operation

A ground segment is used for managing a spacecraft and distributing payload and telemetry data. This ground segment mainly consists of a ground station, mission control center and ground networks. For pre-launch activities it also includes launch facilities and integration and test facilities.

In principle, communication takes place between the control center and the spacecraft over a ground station. The communication between ground and on-board is done with the help of high-frequency signals. However, the communication with the spacecraft is not always possible. Missions in low earth orbit with a typical circulation time of 90 minutes are categorized due to short ground contact times of about 10 minutes. This requires a high degree of automation of activities for configuration of the ground station, the satellite and of the transmission to the ground. Modern earth observation satellites in low orbits typically require 1-2 contacts per day with a ground station for monitoring and control in routine operation. [49]

3 Machine Learning

Defined by A. Samuel in 1959 machine learning referred to as genetic programming are algorithms, which build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task [50]. A shorter version defined by M. Mohri describes it as computational methods using experience to improve performance or to make accurate predictions [51]. The quality and size of this data are crucial to the success of the prediction made by the learner. The goal of machine learning is to design efficient and accurate prediction algorithms. Machine Learning can be divided into three categories: Supervised, unsupervised and reinforcement learning. These will be explained in the following chapter. [51, 52]

3.1 Supervised Learning

The learner receives a set of labeled examples as training data and makes predictions for all unseen points. This is the most common scenario associated with classification and regression. The available data has known labels as output. It involves a supervisor that is more knowledgeable than the neural network itself. For example, the supervisor feeds some example data about which the supervisor already knows the answers, as shown in Figure 11. The supervisor guides the system by tagging the output. A typical example is an email filter that can tag which email is spam or not spam. It has been trained by millions of emails (input data), which have already been classified in spam or not spam [51]. Another example is the recommendation engines of movie streaming services [53].

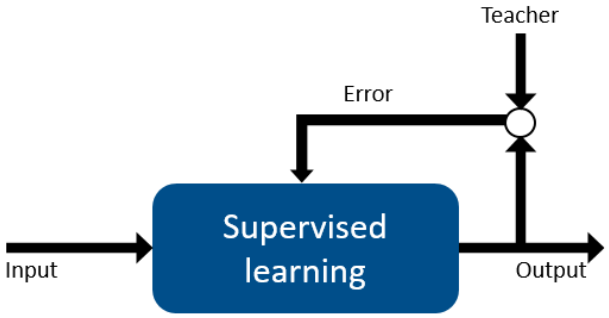


Figure 11: Basic structure of supervised learning [54]

When the data is used to predict a categorical variable, supervised learning is also called classification. This is the case when assigning a label or indicator. For example, document classification consists of assigning a category such as politics, business, sports, or weather to each document. In contrast, image classification consists of assigning to each image a category such as a car, train, or plane. The number of categories in such tasks is often less than a few hundred. However, it can be much more abundant in some problematic tasks and even unbounded as in OCR, text classification, or speech recognition. [51]
Furthermore, supervised learning is often connected to regression. Regression is the problem of predicting a real value for each item. Examples of regression include prediction of stock values or that of variations of economic variables. In regression, the penalty for an incorrect prediction depends on the magnitude of the difference between the true and predicted values, in contrast with the classification problem, where there is typically no notion of closeness between various categories. [51]

Common Algorithms used for Classification and Regression are [55]:

- Support Vector Machines
- Naïve Bayes
- Decision Trees
- Logistic/Linear Regression
- Random Forest

3.2 Unsupervised Learning

The learner exclusively receives unlabeled training data and makes predictions for all unseen points, as shown in Figure 12 [1]. This is useful when there is no example data set with known answers and you are searching for a hidden pattern [2]. Anomaly detection, such as flagging unusual credit card transactions to prevent fraud, is an example of unsupervised learning [53]. Unsupervised learning is mainly used for clustering and association.

Figure 12: Basic structure of unsupervised learning [54]

Clustering mainly handles with finding a structure or pattern in a collection of uncategorized data. Clustering algorithms will process the available data and find natural clusters (groups) if they exist in the data. Clustering is often used to analyze vast data sets. For example, in the context of social network analysis, clustering algorithms attempt to identify natural communities within large groups of people. [51]
Association is a method for discovering interesting relations between variables in large databases. For example, people that buy a new home are most likely to buy new furniture. [56]

Algorithms for Clustering and Association are:

- K-means algorithm
- K-modes algorithm
- Gaussian Mixture Model
- Density-based Spatial Clustering of Applications with Noise (DBSCAN)
- Principal Component Analysis
- Singular Value Decomposition

3.3 Reinforcement Learning

To collect information, the learner actively interacts with the environment and in some cases effects the environment. Afterward, the learner receives an immediate reward for each action, as shown in Figure 13. The goal of the learner is to maximize his reward over the course of actions and iterations with the environment. Applications of reinforcement learning are heavily used in areas such as robotics. For example, to let a robot learn to walk by itself. [54]

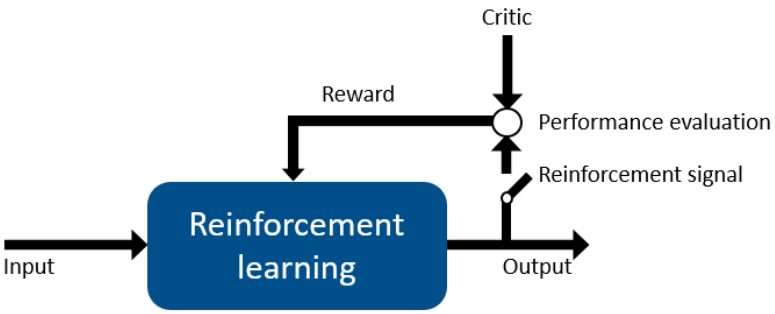


Figure 13: Basic structure of reinforcement learning [54]

Common algorithms are:

- Monte Carlo
- Q-learning
- State–action–reward–state–action (SARSA)

3.4 Deep learning

Deep learning is a subset of machine learning, as visualized in Figure 14. Since around 1995, research of deep learning algorithms has been published [13]. Greater data availability and increasing computational power lead to increased successes of neural networks, and this area was reborn under the new label of deep learning. The basis of deep learning is artificial neural networks and representation learning. Algorithms are Convolutional Neural Networks (CNNs) or Deep Belief Networks and variations of them. Deep learning algorithms automatically learn a good representation of the given input. The architectures of multi-layer neural networks with up to about hundreds of layers are very complex and will be described in more detail in chapter 4.3.4. These algorithms are today enabling many groups to achieve ground-breaking results in vision, speech, language and robotics. [57]

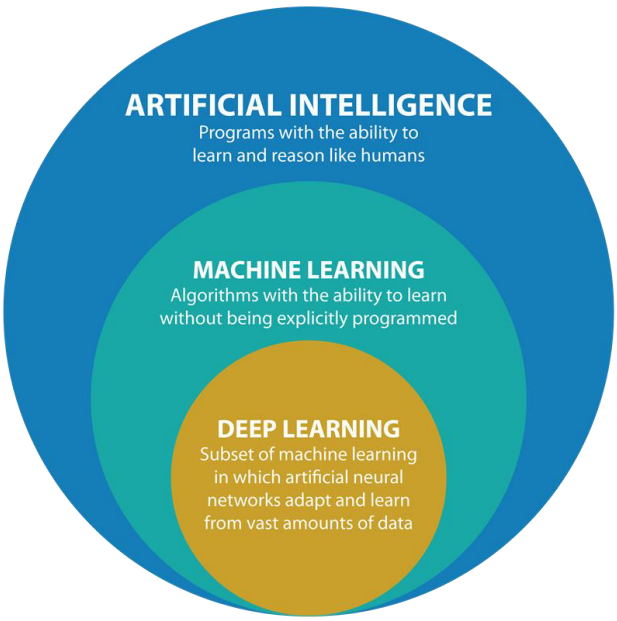


Figure 14: Deep learning in the context of artificial intelligence [58]

4 Scenarios

Another research field with high impact expected on the engineering process of production systems is artificial intelligence and mainly its subfield, machine learning. Improvements in the accessibility of computing power lead to a variety of applications in Industry 4.0. In the following chapter, three use cases will be described: Predictive Maintenance, Self-Learning Production System and Computer Vision. Recent developments in the field of AI made it possible to produce several frameworks and libraries or other program suits. The available open-source libraries, algorithms or commercial software suits for each use case will be described.

4.1 Predictive Maintenance

Predictive Maintenance is based on condition monitoring. Therefore, condition monitoring will be described first. Condition monitoring is the monitoring of various parameters in a production line to identify potential failures [59]. Common parameters are vibration or temperature, which are analyzed in real-time. Condition-based maintenance takes into account the usage conditions of the equipment better than the traditional predetermined maintenance. Corrective maintenance is defined as a "maintenance carried out after fault recognition and intended to put an item into a state in which it can perform a required function." [60]. Because this type of maintenance is not relevant for this thesis it will not be further analyzed. To categorize the terms, an overview of all maintenance terms is shown in Figure 15.

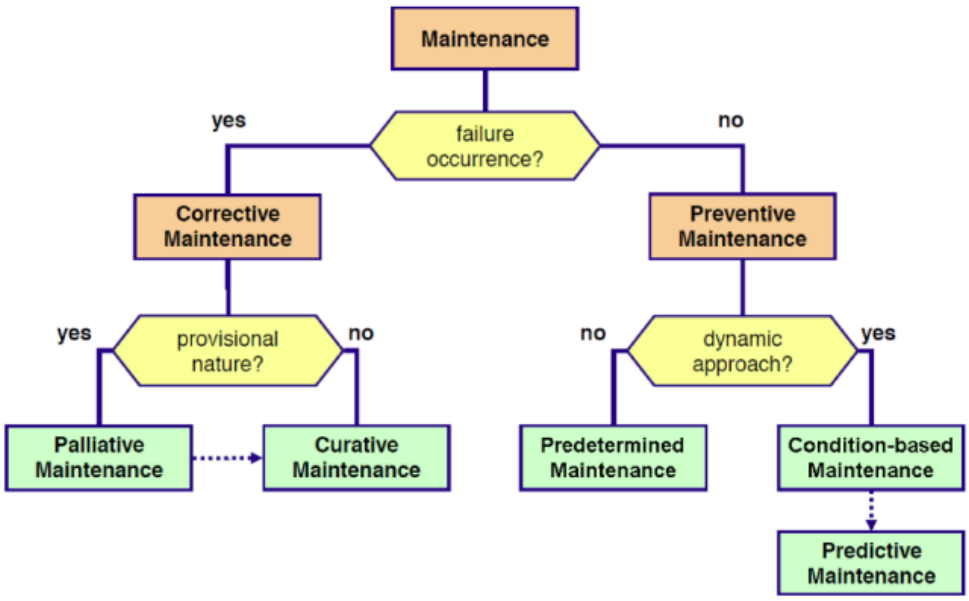


Figure 15: Forms of maintenance according to DIN EN 13306 [60]

Predictive Maintenance is defined as "a condition-based maintenance carried out following a forecast derived from the analysis and evaluation of the significant parameters of the degradation of the item." [60]. The goal to increase the availability of machines by recognizing failures or problems before significant damage is caused to the equipment.

According to R. Gouriveau the benefits of Predictive Maintenance are [59]:

- Reduction of the number of breakdowns
- Increased reliability of production processes
- Improvement of personnel safety and company image
- Reduction of periods of inactivity for the equipment (costly)
- Increment of the performance of the company

4.1.1 Algorithm development

The development of a Predictive Maintenance system is complex and involves many different steps, which will be described in the following. Figure 16 visualizes the workflow of these steps until a successful prediction model is built.

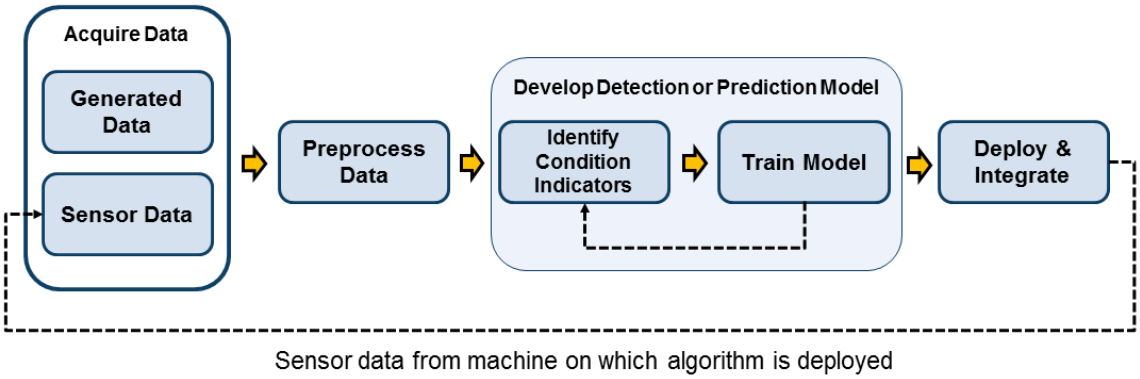


Figure 16: Workflow for developing a Predictive Maintenance algorithm [61]

The first requirement for a Predictive Maintenance approach is a large volume of data in order for machine learning algorithms to learn typical failure patterns. This data is collected from serval sensors in the industrial machines and has to represent healthy and faulty operations. The data should also be collected under different operating conditions. Another approach is if there is not enough data available to generate a mathematical model of the machine and estimate its parameters from the sensor data. Then the model can simulate different fault states under different fault conditions to generate failure data. Both simulated and real data can be used to develop an algorithm.
The next step is to preprocess the data. Remove potential outliers and filter out the noise. Sometimes it is also necessary to do further preprocessing. For example, converting time domain data to frequency domain can help to identify useful features, also called condition indicators. These are used to distinguish healthy from faulty condition. Afterwards these condition indicators are used to train machine learning models. These machine learning models can detect anomalies or identify fault types, which part needs to be fixed. Furthermore, it can also predict the remaining useful life of the machine. After developing the algorithm, it can be deployed in the cloud or on-site with edge devices. If there is a huge amount of data to be transmitted, it is also possible to run the steps preprocess data and identify condition indicators on the edge devices. And only send the extracted features to the cloud. [61]
Several different approaches, which type of machine learning algorithm to use, can be found in the thesis of P. Jahnke [62]. Over 12 possible machine learning techniques show promising use. Because not all can be investigated in this thesis, it will be mainly focused on supervised learning algorithms which will be presented in the following chapters. According to P. Jahnke Support Vector Machine is one of the most used technique in recent literature.

4.1.2 Decision tree learning

Decision tree learning is one type of supervised learning used for classification and regression. As the name indicates, it uses a decision tree as a predictive model. Decision trees are used to classify an object to a predefined set of classes based on their attributes. A decision tree consists of the following elements: Root Node is the starting point of the tree. It represents the entire sample and gets divided into two or more sets. Splitting is the process of dividing a node into two or more sub-nodes. Decision Node is a sub-node that splits further. Leaf or Terminal Node is a node that does not split any further. [63, 64]

A decision tree used for predicting the outcome of the class to which the data belongs is called a classification tree [63]. As an example, a simple classification tree is visualized in Figure 17. It is built to analyze whether maintenance has to be performed (Yes) or not (No). These are the two possible classes. The decision tree is executed from top to bottom. The start point is the Root Node (Equipment Age). The lines are the Splitting process with the defined condition (e.g., ≤ 7000 h). If the next step ends on a Terminal Node (Yes or No), the decision is made.

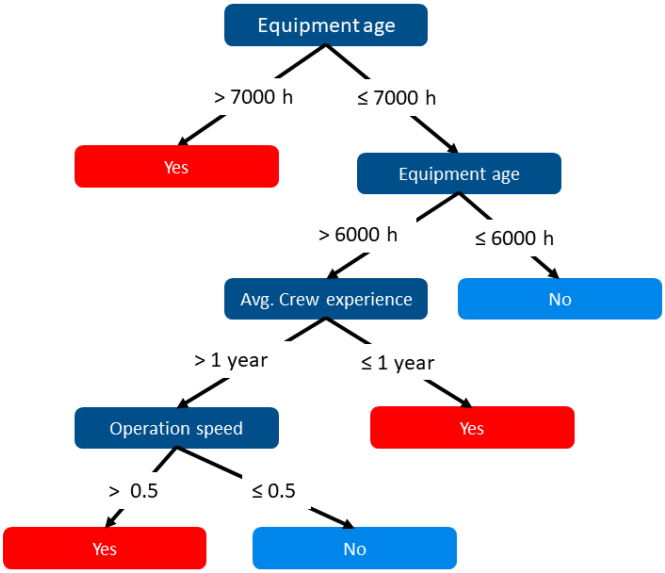


Figure 17: Example of a simple decision tree (based on [65])

A decision tree can help in finding and selecting the right model for a given failure type detection and Predictive Maintenance issue [62]. Building on this knowledge, the use of gradient boosting or in general boosting is a useful technique for applying on big datasets according to K. Liulys [66]. Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. Boosting is a procedure that establishes an efficient decision rule for a classification problem by combining several simple rules. These rules are called weak classifier (weak learner). The result of boosting is defined as a strong classifier (strong learner) [67]. Using this technique, subsequent predictors learn from the mistakes of the previous predictors. Because new predictors are learning from mistakes committed by previous predictors, it takes less time or iterations to reach close to the actual predictions.

4.1.3 Support Vector Machine

Support Vector Machines (SVM) are a set of supervised learning methods used for classification, regression and outlier detection. Given a set of training examples, each labeled to one of two classes. A SVM training algorithm builds a model that assigns new examples to one of the classes. An example of a linear Support Vector Classifier (SVC) is shown in Figure 18. With the two classes of blue and red data points.

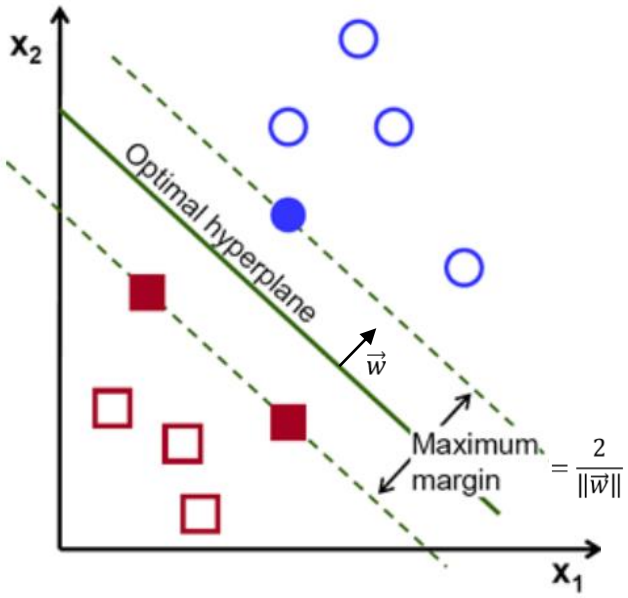


Figure 18: Linear Support Vector Classifier [68]

The goal of an SVM is to find a line that has the maximum margin, e.g., the maximum distance between data points of both classes. This line is called a hyperplane. Data points on one side of the hyperplane will be classified to a specific class while data points on the other side of the hyperplane will be classified to a different class (e.g., blue or red). For a linear SVC two parallel hyperplanes can be selected to separate the two classes of data, so that the distance between them is as large as possible. The region bounded by these two hyperplanes is called the margin. The maximum margin hyperplane is the hyperplane that lies halfway between them. Mathematically a linear SVC is defined the following [69]:

With a given training dataset of n points

$$(\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n)$$

where y_i are either 1 or -1 to indicate which class is belongs. The objective is to find the maximum hyperplane that divides the group of points $\vec{x}_i$ for the classes $y_i \in \{-1, +1\}$. So that the distance between the hyperplane and the nearest point $\vec{x}_i$ from either group is maximized. A hyperplane is defined as:

$$\vec{w} * \vec{x} - b = 0$$

Where $\vec{w}$ is the normal vector to the hyperplane. Theoretically, there is an infinite number of hyperplanes that separate both classes. The distance between two hyperplanes is $\frac{2}{\|\vec{w}\|}$, so to maximize the distance between the planes $\|\vec{w}\|$ needs to be minimized. With the following constraints:

$$y_i(\vec{w} * \vec{x}_i - b) \geq 1, \quad for\ all\ 1 \leq i \leq n$$

Nonlinear classification is also possible with Support Vector Machines by applying the kernel method [70]. Supported kernel functions in scikit-learn are: linear, polynomial, Radial Basis Function (RBF), sigmoid [71]. Figure 19 shows how to plot the decision surface for four SVM classifiers with different kernels. The shapes of the boundaries depend on the kind of kernel and its parameters.

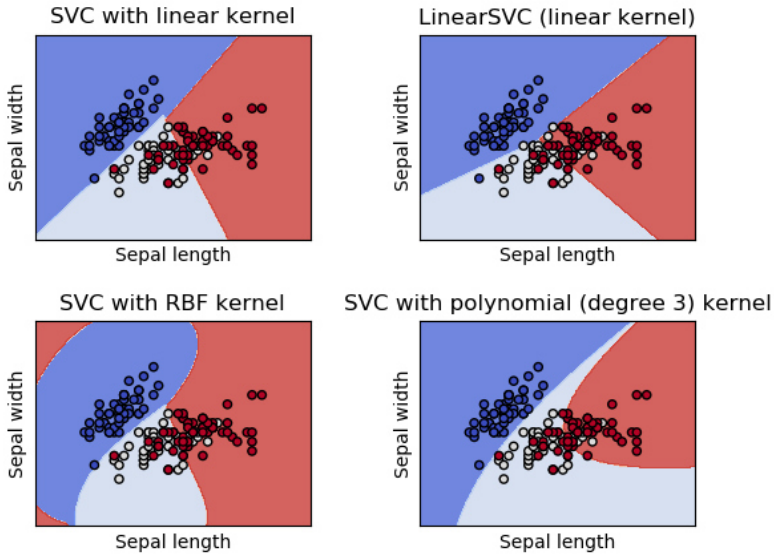


Figure 19: Four SVM classifiers with different kernels [71]

4.1.4 Remaining useful life

The Remaining Useful Life (RUL) of a system is defined as the length from the current time to the end of the useful life. In the context of Predictive Maintenance, it is the amount of time that an asset is operational before the next failure occurs. Predicting the RUL from system data is a major goal in Predictive Maintenance [61]. In conventional data-based approaches, life characteristics of identical systems and lifetime data are required for estimating the RUL. Such information is often rare in reality or even nonexistent, because they are too costly or time-consuming to collect. Data needed for the prediction can be sensor data or log data, which values changes predictably as the system degrades and operates in different settings. For example, the vibration increases because of a worn-out bearing. Datasets can be lifetime data indicating how much time passed until a similar machine reaches a failure. Run-to-failure histories of similar machines or a condition indicator that detects the failure at a known threshold value can also be applied [72]. Based on these datasets, regression models can be used to calculate the remaining useful life. If a run-to-failure data was not recorded or available, it is possible to describe the failure with a threshold value. An example would be that the temperature of a pump liquid cannot exceed 80 °C. This process of failure detection through threshold is visualized in Figure 20. [73]

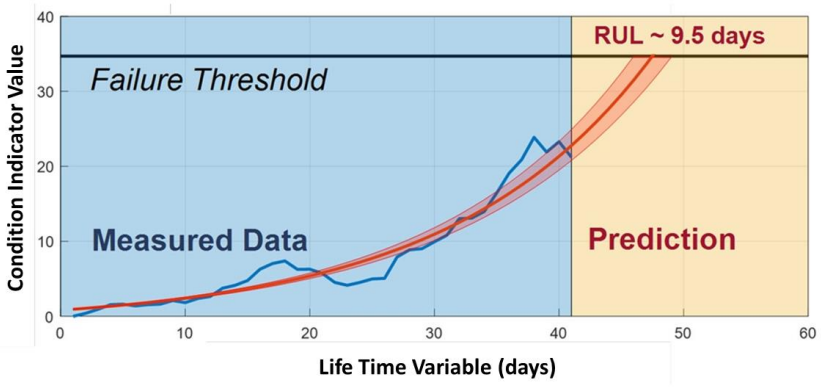


Figure 20: Predictive curve of the health condition of a system [72]

The measured data is the blue plot. It is calculated from different sensors, which can represent a failure. They are summarized as the condition indicator. They are time depended and rise if the condition of the system is deteriorating. The regression algorithm predicts the course of the future condition state (red line). The threshold in this case is set to 35 and the remaining useful life is calculated for the time the predictive plot passes the defined threshold. In this case, the system will probably fail 9.5 days after the last measurement. [72]

4.1.5 Comparison of Predictive Maintenance platforms

There are several different possible approaches to implement Predictive Maintenance in the Space Factory 4.0 framework. There are already commercial toolboxes available or open source solutions, which will be described in this chapter.

The “Predictive Maintenance Toolbox” integrated in MATLAB is a commercial available software. It provides the most common algorithms, which have been described in the previous chapter. It can estimate the Remaining Useful Life, use classification models for fault diagnoses and different filters for anomaly detection. It has a data management system and labeling system to organize files locally or in the cloud. Furthermore, it can also simulate failure data with Simulink, where the digital twin simulates the behavior of the real-world machine. [74]

Another commercial software is MindSphere, which is a cloud-based IoT open operating system from Siemens. It is actually designed to connect all products, plants and machines in a factory to harness all data. However, it also offers analytical tools for Predictive Maintenance. Currently, the digital twins of the Space Factory 4.0 are modeled with Siemens NX and with a combination of Teamcenter as a Product-Lifecycle Management, the integration in MindSphere is convenient. [75]

Microsoft Azure Predictive Maintenance platform is half commercial half open source. Azure provides algorithms and templates open source. But on the other hand, the commercial Microsoft Azure AI platform is necessary to run these applications in the cloud. It provides an extensive guide on the data science process, including the steps of data preparation, feature engineering, model creation, and model operationalization. [76]

Scikit-learn is not an out-of-the-box Predictive Maintenance tool, but it is entirely open source. It is machine learning library for the programming language Python. It provides all necessary machine learning algorithm necessary for a Predictive Maintenance application. It features various classification, regression and clustering algorithm to analyze data. This approach was chosen for an application described in chapter 5.5. [71]

4.2 Self-Learning Production Systems

The fundamental concept of a Self-Learning Production System involves an evolvement of the industrial system over a certain time. The capacity to evolve is the ability of a system to change its behavior, i.e., its internal status or parameters, depending on different production situations [77]. There are different approaches possible to achieve self-learning, but this chapter will concentrate on the machine learning method reinforcement learning. Furthermore, different projects will be introduced, where reinforcement learning was successfully implemented.

4.2.1 Reinforcement Learning

Reinforcement learning is a method of machine learning wherein the software agent learns to perform specific actions in an environment which lead it to maximum reward. It does so by exploration and exploitation of knowledge it learns by repeated trials of maximizing the reward. [78]

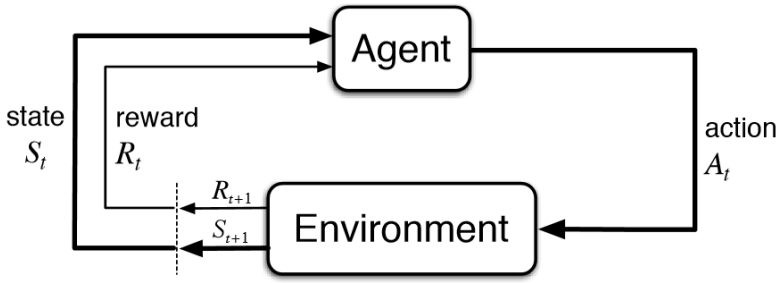


Figure 21: Basic diagram of a reinforcement learning scenario [78]

Typically, a reinforcement learning scenario is composed of two components, an agent and an environment, as seen in Figure 21. The learner and decision-maker is called the agent. The thing it interacts with, comprising everything outside the agent, is called the environment. These two elements interact continually, the agent selecting actions and the environment responding to those actions and presenting new situations to the agent. The environment also gives rise to rewards, special numerical values that the agent tries to maximize. A complete specification of an environment defines a task, one instance of the reinforcement learning problem. At each time-step t, the agent observes S_t and selects an action A_t. The environment responds with a reward signal R_{t+1} and a new state S_{t+1}. An intuitive way to understand the relation between the agent and its environment is with the following example: [78]

Environment:	You are in state 65. You have 4 possible actions.
Agent:	I will take action 2.
Environment:	You received a reward of 7 units. You are now in state 15. You have 2 possible actions.
Agent:	I will take action 1.
Environment:	You received a reward of -4 units. You are now in state 65. You have 4 possible actions.
Agent:	I will take action 2.
Environment:	You received a reward of 5 units. You are now in state 44. You have 5 possible actions.
⋮	⋮

The particular states and actions vary greatly from task to task, and how they are represented can strongly affect performance. In reinforcement learning, as in other kinds of learning, such representational choices are at present more art than science. [78]

4.2.2 Algorithms

Model-free algorithms rely on trial and error to update their knowledge. They do not require space to store all the combinations of states and actions. Common algorithms are:

- Markov Decision Processes (MDPs)

- Q-Learning

- State-Action-Reward-State-Action (SARSA)

- Deep Q Network (DQNs)

- Deep Deterministic Policy Gradient (DDPG)

An example described in [78] is a Pick-and-Place Robot using reinforcement learning to control the motion of a robot arm in a repetitive pick-and-place task. If the goal of the robot is to learn movements that are fast and smooth, the learning agent will have to control the motors directly and have low-latency information about the current positions and velocities of the mechanical linkages. The actions, in this case, might be the voltages applied to each motor at each joint, and the states might be the latest readings of joint angles and velocities. The reward might be +1 for each object successfully picked up and placed. To encourage smooth movements, on each time step, a small, negative reward can be given as a function of the moment-to-moment “jerkiness" of the motion. This simple example can be transferred to different Industry 4.0 use cases. The project InPuls (Intelligente Produktionsprozesse und lernfähige Systeme in KMUs) aims to achieve a flexible automation even in the event of unforeseeable conditions. It is funded by the Verband Deutscher Maschinen- und Anlagenbauer (VDMA) and the use of artificial intelligence for process control is being investigated. [79]

4.2.3 InPuls

Different machine learning methods are useful for varying complex scenarios. In the industrial production, three categories have high potential: process monitoring, process optimization, process control, which increase in complexity in ascending order. Table 5 gives an overview of these machine learning methods and resources needed. Process monitoring is the first step, collecting data form different data sources. No machine learning algorithm are applied and the data is only visualized and statistical methods are used. Higher complexity is reached with process optimization. It includes all necessities from process monitoring and enhances them with supervised and unsupervised learning tools. Reinforcement learning is the most complex approach and can be used for process control.

Table 5: Classification of reinforcement learning in automation [79]

	Process monitoring	**Process optimization**	**Process control**
Functions	Situational awareness, predictive information	Planning, decision support	Automated response to changes in environment
Advantages	Enhanced quality, reduced downtimes, reduced shortfalls	Enhanced efficiency, improved use, bigger yields, effective design	Enhanced production/ productivity, lower labour costs, reduced scrap
Resources	Data sources (e.g. connected sensors)	Process monitoring + sophisticated analyse tools	Process optimization + integration of physical Systems (robots)
Methods	Visualisation and descriptive statistics	Supervised and unsupervised learning	Reinforcement learning

Complexity

The number of sensors used in the production plant is increasing. With the help of these sensors, the current status of the plant can be monitored or already be used for minor predictions of the future state. These technologies offer an increased process quality through better monitoring, reduced downtimes and increased process reliability. This process monitoring can usually be realized with simple statistical methods. Further improvement can be achieved with the help of machine learning in process optimization. Especially in form of increased efficiency and cost reduction. In process optimization methods from the field of supervised and unsupervised learning are used, for example, for planning or as a decision support. The complexity increases further if the process has to be controlled using machine learning methods, since optimization and execution on the physical system are very closely intertwined. With the methods of supervised and unsupervised learning, one reaches the boundaries of what is possible. In this field, reinforcement learning is promising, but only little researched or implemented. The independent learning of an intelligent control strategy has enormous potential, as it enables process optimization without modeling and therefore high autonomous flexibility. [79]

Reinforcement learning includes the following potential:

- Development of complex control strategies that are independent of expert knowledge.
- Find completely new solutions for known control problems
- New solution can be more efficient than conventional strategies
- Ability to determine a control strategy in real-time, which would be too computationally intensive using a simulation
- New motion strategies in robotic
- Flexibility of path planning and motion execution

4.2.4 Applications

One project in the InPuls framework is the “Cense2.0 - Cognition Enhanced Self-Optimization” [80]. It consists of a reinforcement learning based, cognition-enhanced six-axis industrial robot for complex motion planning along continuous trajectories, as, e.g., needed for welding, gluing or cutting processes in production. The prototype demonstrator is inspired by the classic wire loop game, which involves guiding a metal loop along the path of a curved wire from start to finish while avoiding any contact between the wire and the loop. The work shows that the reinforcement learning agent is capable of learning how to control the robot to successfully play the wire loop game without the need of modelling the wire or programming the robot motion beforehand. Furthermore, the extension of the system by a visual sensor (a camera) allows the agent to sufficiently generalize the learning problem so that it can solve new or reshaped wires without the need for additional learning. [81, 82]

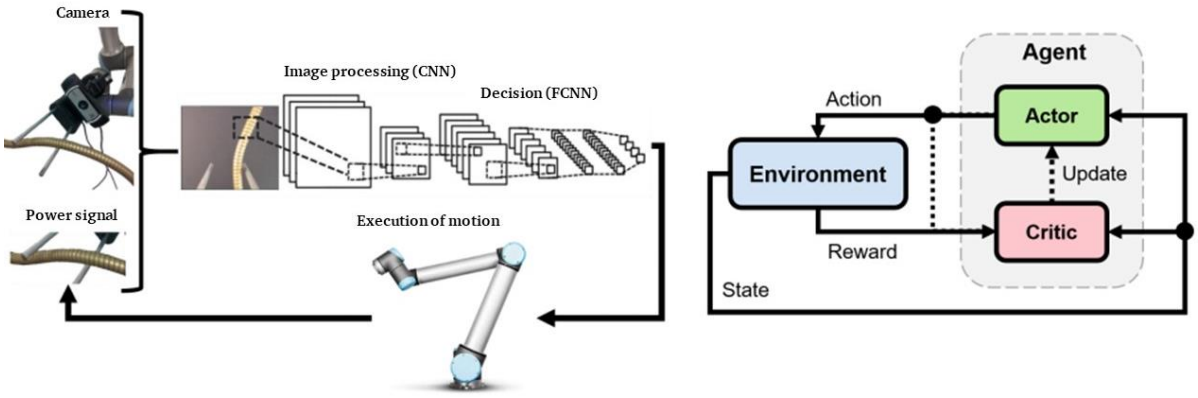


Figure 22: Process of Cense 2.0 [80]

Another similar project is a robot, which is trained by reinforcement learning for an autonomous force-controlled assembly. Joining tasks are very contact rich and therefore represent a complex learning task, which often cannot be learned in simulations. In industry, these so-called peg-in-hole tasks often require a higher positioning accuracy than it is possible with current robots [82]. In the paper, the use of Guided Policy Search (GPS) is researched, which is a model-based reinforcement learning approach that allows for learning manipulation skills in a continuous state and action space [83]. The scenario was performed with a hole size of 20 mm diameter and different pin sizes to test the precision of the assembly. With the used method, a precision of 2 mm could be achieved [79]. The long-time goal of the project is to create an autonomous assembly cell for unplanned assembly situations. A robot independently learns a suitable assembly movement without having to describe the exact kinematic and dynamic conditions of the gripping system and component.

The training of a reinforcement learning model requires a certain amount of resources. These resources, often called training costs, should be kept as low as possible. The most crucial cost factor, according to VDMA, is the time required to train a reinforcement learning algorithm, large amounts of data are generated by testing the real process [79]. This requires that the real process is short and can be repeated frequently. Therefore another approach is “Collective Robot Reinforcement Learning with Distributed Asynchronous Guided Policy Search” [84]. In this case, multiple robots can share their experience with one another, and thereby, learn a policy collectively. A distributed and asynchronous version of Guided Policy Search is used and demonstrates collective policy learning on a vision-based door opening task using four robots. With this method, a better generalization, utilization, and training times than the single robot alternative can be achieved.

4.2.5 Motion planning

In this chapter the motion planning with reinforcement learning will be introduced. As described in chapter 4.2.1 classical reinforcement learning approaches are based on the assumption that we have a Markov Decision Process consisting of the set of states S, set of actions A and the rewards R that capture the dynamics of a system. Q-learning is a paradigm that can be used to allow an agent to find an optimal policy for choosing an action for any given finite Markov Decision Process [81]. In general, a policy is a deliberate system of principles to guide decisions or more specifically, a decision function that specifies what the agent will do for each possible value that it can sense [85]. Q-learning allows different policies that can be changed during training. Q-learning is based on the existence of the Q-function, which models the quality of an action a which is performed in state s at time step t.

$$Q(s_t, a_t) = \max\{R_{t+1}\}$$

For an example of the learning phase of a robot the first example from chapter 4.2.4 game is chosen. It is the experiment based on the wire loop game. The agent explores its environment and stores its experiences as state-action-reward triples (s, a, r) and iteratively estimates the Q-values for each state-action pair it encounters. The agent starts with no experience about the environment at all and will perform random actions every time it finds itself in an unknown state initializing the corresponding Q-Values at the same time. As experiences are gathered over time, the agent starts to update the Q-values and learns how to progress along the wire while avoiding any contact between the loop and the wire. A general problem is that the agent's learned strategy tends towards a local optimum. It is called the exploration-exploitation dilemma [86]. This problem can be solved by the so-called ϵ-greedy exploration. An example of the algorithm used in the wire loop games is shown in Figure 23. The ϵ-greedy exploration enforces that the agent chooses a random action with probability ϵ or the greedy action with probability $1 - \epsilon$. [81, 87]

initialize Q-values arbitrarily
observe initial state s
set initial value for ϵ, e.g. $\epsilon_{init} = 0.9$
repeat
  select action a
    with probability ϵ select a random action a
    otherwise select $a = argmax_{a'}\{Q(s, a')\}$
  perform action a
  observe reward r and new state s'
  update Q-value
  update state: $s = s'$
until terminated
if $\epsilon > \epsilon_{lim} = 0.1$, **then**
  decrease ϵ, e.g. $\epsilon = \epsilon - (1/number\ of\ episodes)$
end if

Figure 23: Q-learning algorithm with ϵ-greedy exploration [81]

4.3 Computer Vision

Computer Vision is one primary function in this thesis. Therefore, its fundamentals will be described deeply and a distinction is made between different vision systems. Later on, neural networks will be introduced and explained how Convolutional Neural Networks can be used for object detection.

4.3.1 Definition of Computer Vision

Computer Vision is the process whereby a machine, usually a digital computer, automatically processes an image and tries to understand what is in the image, like a human visual system would be able to do. That means it recognizes the content of the image. For example, the content may be a machined part, and the objective maybe not only to locate the part but to inspect it as well. [88]
As shown in Figure 24, the Computer Vision system accepts an input that is an image and produces a number of measurements extracted from the image. These measurements are called features, which are used by the pattern recognition system to make a decision.

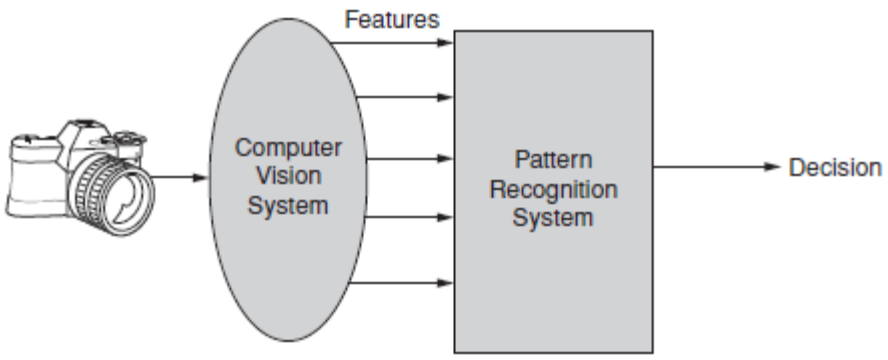


Figure 24: Computer Vision System and Pattern Recognition System [88]

There are several different terms used in the literature, such as Image Processing, Machine Vision and Image Understanding. For example, the pattern recognition system must decide whether an object is an axe or a hammer, and it receives as input the length of the handle and the mass of the head. If the length of the handle is to be determined from an image, the system that makes that determination is a Computer Vision system. So the Computer Vision system makes the measurements and transfers it to the pattern recognition system. Therefore, it is also often described as rule-based machine vision. Another example is the current research in deep learning, which uses example-based algorithms and neural networks to analyze an image. [88, 89]

4.3.2 Rule-based machine vision

On a production line, a rule-based machine vision system can inspect hundreds, or even thousands of parts per minute with high accuracy. It is more cost-effective than human inspection. The output of that visual data is based on a programmatic, rule-based approach solving inspection problems. On a factory floor, traditional rule-based machine vision is ideal for: guidance (position, orientation), identification (barcodes, data matrix codes, marks, characters), gauging (comparison of distances with specified values), inspection (flaws, broken part). Rule-based machine vision is excellent with a known set of variables: Is a part present or absent? Exactly how far apart is this object from the other? Where does this robot need to pick up this part? These jobs are easy to deploy on the assembly line in a controlled environment.

4.3.3 Deep learning

If there is not a controlled environment, deep learning can be applied. It solves vision applications too difficult to program with rule-based algorithms. It uses complex neural networks to analyze the images. A typical industrial example is that a manufacturer wants to detect defects in the products they make. One way to approach this would be via traditional machine vision. With traditional machine vision, engineers would have to explicitly program the inspection to account for the millions of variations that could happen: the size and type of defect and the location of the defects. That becomes a very time-consuming application to both maintain and program because of the inherent variations. With a deep learning based approach, the algorithm takes the examples provided by the user and automatically creates an understanding of the part being inspected. By creating an inspection that learns what a good part looks like, even accounting for slight variations, the solution can then flag when something looks amiss, such as scratches, foreign objects or other visual defects. Users can then improve the solution by providing more data for the tool, so the learning process increases. The more data the deep learning application has, the better it will perform over time to spot anomalies. [90]

According to James Furbush there are five steps to consider before deploying a deep learning project [89]:

1. Set the proper expectations
2. Understand deep learning's return on investment
3. Master the resource planning and needs
4. Start small with an initial pilot project
5. Undergo a phased project approach

Some benefits of a deep learning based vision system are listed below [53]:

- Solve vision applications too difficult to program with rule-based algorithms
- Handle confusing backgrounds and variations in part appearance
- Maintain applications and re-train with new image data on the factory floor
- Adapt to new examples without re-programming core networks

The highest goal of a vision system is always to outperform the human inspector. The human vision system can understand correlations in an image effortless, as well as largely independent of viewpoint and object orientation. The ability to interpret and image like a human became possible since recent leaps in deep learning. A human inspector’s abilities decrease over time. A deep learning system in contrast can be more consistent, reliable and faster. One major disadvantage of the traditional machine vision is that complex visual understanding rise in programming effort. Deep learning applications are easy to configure, even so, a large dataset is needed and can understand complex correlations in an image. A deep learning system can also improve over time if new data will be imputed. An overview of benefits compared to a human inspector and rule-based machine vision is shown in Table 6. [53]

Table 6: Comparison of Deep Learning, Human Inspector and Rule-based Machine Vision [53]

Deep Learning vs. Human Inspector	Deep Learning vs. Rule-based Machine Vision
More consistent Reduces inconsistencies between different human inspectors	**Designed for complex applications** Solves complex inspection and classification applications impossible or difficult with classic rule-based algorithms
More reliable Operates more reliably even when scaled or reproduced to other lines	**Less programming effort** Applications can be set up quickly, speeding up proof of concept and development
Faster Identifies defects in milliseconds, supporting high-speed applications and improving throughput	**Tolerates variations** Handles defect variations for applications that require an appreciation of acceptable deviations from the control

4.3.4 Neural Network

The main element of a deep learning system is the neural network, which will be described in the following chapters.

As defined in [91] a neuron is a nonlinear bounded function $y = f(x_1, x_2, \dots x_n; w_1, w_2 \dots, w_p)$. Where the x_i are the variables and the w_j are the parameters (or weights) of the neuron. The function used in a neuron is generally termed as an activation function. Five common activation function used are: step, sigmoid, tanh, ReLU and leaky ReLU. In this thesis only ReLU will be described.

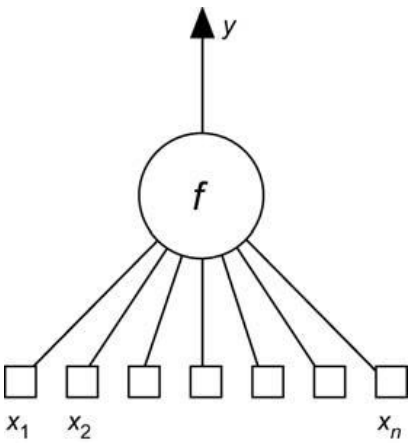


Figure 25: Neuron

The Rectified Linear Unit (ReLU) is the most commonly used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive value x, it returns that value back. So, it can be written as $f(x) = \max(0, x)$.

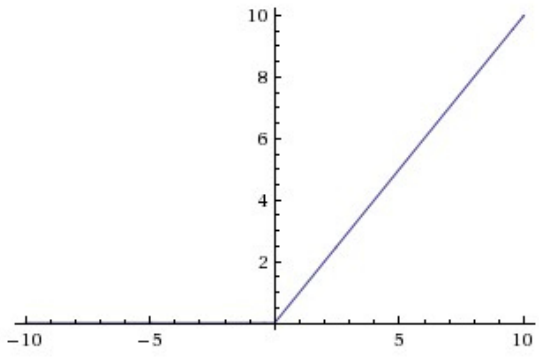


Figure 26: Rectified Linear Unit activation function, which is zero when x<0 and then linear with slope 1 when x > 0.

Neural Networks are modeled as collections of neurons that are connected in an acyclic graph. The outputs of some neurons can become inputs to other neurons. Cycles are not allowed since that would imply an infinite loop in the forward pass of a network. Neural Network models are often organized into distinct layers of neurons.

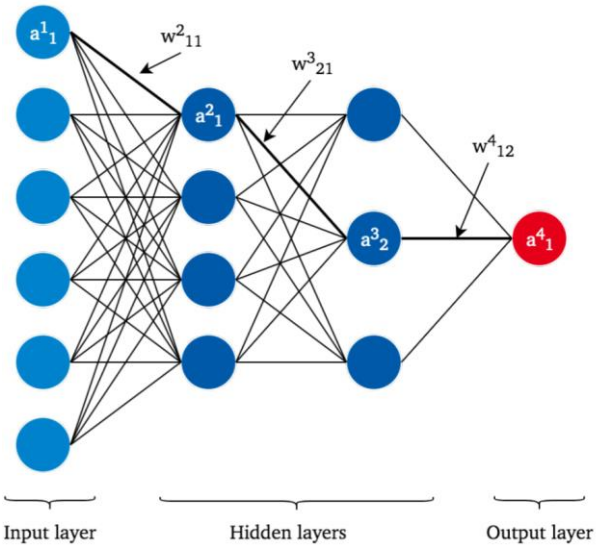


Figure 27: Fully connected neural network (based on [91])

The structure of neural network consists of nodes/units (a_j^l), which are organized as layers. The single layers of a neural network are wired together via edges that are called weights (w_{jk}^l). This subsection will focus on the most basic architecture of a neural network, called a feed-forward network. For regular neural networks, the most common layer type is the fully-connected layer in which neurons between two adjacent layers are fully pairwise connected, but neurons within a single layer share no connections [92]. This type of neural network is shown in Figure 27. It contains an input layer with six units, two hidden layers with four respectively three units and an output layer containing on a single unit. In this case, hidden layers are all layers between input and output layer.

As shown in Figure 27 a_j^l will refer to the j^{th} unit in the l^{th} layer. Weight w_{jk}^l stands for the weight which connects the k^{th} unit in the $(l-1)^{th}$ layer with the j^{th} unit in the l^{th} layer. During the forward propagation process, the neural network calculates an output for a specific input with respect to the current weights in the network. Thus, the network can be written as a

function $f(x, w)$ where x is an input vector or matrix and w is the set of all weights for this network. The units in the input layer receive their value from the input vector. All unit in the following layers are calculated using this equation called activation function [93]:

$$a_j^l = \sigma(\sum_k w_{jk}^l a_k^{l-1} + b_j^l)$$

There are different kinds of neural networks as Feedforward Neural Networks (FNN), Recurrent Neural Networks (RNN), Multilayer Perceptron (MLP), Radial Basis Function Networks (RBF). In this thesis, the use of Convolutional Neural Networks will be investigated. Hence, these will be described fully in the next chapter.

4.3.5 Convolutional Neural Networks (CNN)

Convolutional Neural Networks are designed to work with grid-structured inputs, which have strong spatial dependencies in local regions of the grid. The most obvious example of grid-structured data is a 2-dimensional image. Therefore, Convolutional Neural Networks are mainly used in the field of Computer Vision, but it can be used for all types of temporal, spatial, and spatiotemporal data. It derives its name from the type of hidden layers it consists of. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers, and normalization layers. A convolution is a dot-product operation between a grid-structured set of weights and similar grid-structured inputs drawn from different spatial localities in the input volume. Unlike a regular Neural Network (shown in 4.3.4) the layers of a CNN have neurons arranged in 3 dimensions: width, height, depth [94]. The depth of a layer in a convolutional neural network should not be confused with the depth of the full network itself. When used in this context, “depth” refers to the number of channels in each layer, e.g., number of primary color channels (red, green, blue). An example is shown in Figure 28. Every layer of a CNN transforms the 3D input volume to a 3D output volume of neuron activations. The input is a picture with spatial dimensions of 32x32 pixels (width, height) and the depth is 3 (red, green, blue). Therefore, the overall number of pixels is 32x32x3. The Convolutional layer’s parameters consist of a set of learnable filters. In this example it has the size of 5x5x3. The convolution operation slides each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. As we slide the filter over the width and height of the input volume we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. [57, 95]

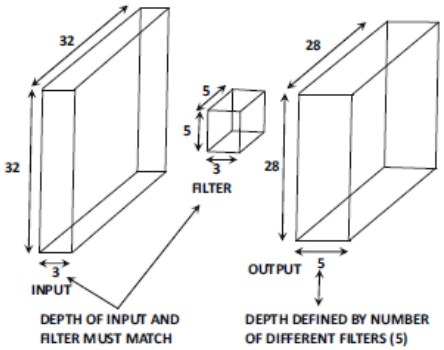


Figure 28: Visualization of one convolutional operation [57]

The depth of the resulting output depends on the number of distinct filters and not on the dimensions of the input layer or filter. Mathematically a convolution can be described with the input I and the filter F with the number of elements N for a 1D image as:

$$F * I(x) = \sum_{i=-N}^{N} F(i)I(x-i)$$

Numerous convolutions are performed on the input, where each operation uses a different filter. This results in different so-called feature maps. In the end, all of these feature maps are summarized together as the final output of the convolution layer. As described in the chapter 4.3.4 Neural Network, we use an activation function to make our output non-linear. In the case of a Convolutional Neural Network, the output of the convolution will be passed through the activation function. This could be the ReLU activation function. After a convolution layer, a pooling layer is added in between CNN layers. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. This shortens the training time and controls overfitting. The most frequent type of pooling is max pooling, which takes the maximum value in each window. It is shown in Figure 29. These window sizes need to be specified beforehand. This decreases the feature map size while at the same time keeping the significant information. [57]

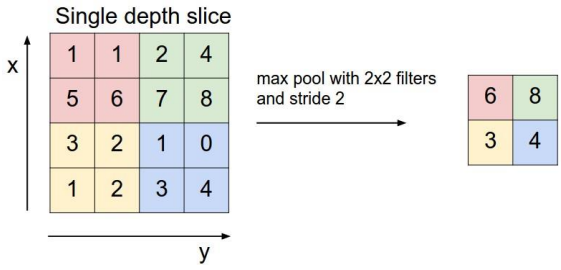


Figure 29: Max Pooling [92]

Stride is the size of the step the convolution filter moves each time. A stride size is usually 1, meaning the filter slides pixel by pixel. By increasing the stride size, the filter is sliding over the input with a larger interval and thus has less overlap between the cells. Padding is used because the size of the feature map is always smaller than the input. To prevent the feature from shrinking a layer of zero-value pixels is added to surround the input with zeros. Additionally, it improves performance because without padding the information at the border would be erased too quickly. [94]

When using a CNN, there are four main parameters to decide on:

- the kernel size (e.g., 5x5x3)
- the filter count (number of filters used)
- stride (the steps of the filter)
- padding

To summarize, a CNN consists mainly of a feature extraction part with the hidden layers and a classification part, as shown in Figure 30. New developed CNNs have hundreds of hidden layers that each learn to detect different features in an image. For example, the first layer detects edges and last detect more complex shapes. The final layer connects every neuron from the last hidden layer to the output neurons. Then the final output is created. After this feature extraction part, the classification consists of a fully connected layer. It then outputs a probability to which class the image belongs. [94]

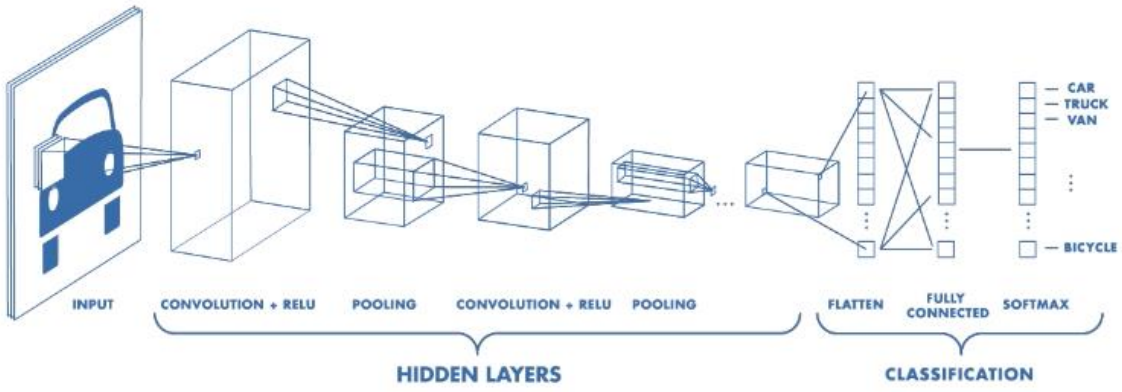


Figure 30: Architecture of a Convolutional Neural Network [96]

To train a CNN from scratch is highly accurate but also very challenging, because significant computational resources are necessary. Another common method relies on transfer learning, which is based on the idea that the knowledge of one type of problem is used to solve a similar problem. For example, a CNN model that has been trained to recognize animals can be used to initialize and train a new model that differentiates between cars and trucks [96]. There are already several CNNs developed, which will be shown in the next chapter.

4.3.6 Comparison of object detection models

The ImageNet challenge is a competition, where research teams evaluate their algorithms on a given data set. To date, the data set consists of more than 14 million labeled images from 20.000 categories. The contest is held annually since 2010. The goal is to compete to achieve higher accuracy on several visual recognition tasks. At the beginning of the contest, the error rate was 28 %. The University of Toronto's 8-layer CNN AlexNet reduced this figure to 16 % in 2012. All subsequent winners were Convolutional Neural Networks. Three major variants (VGG, Inception, ResNet) were created, reducing the error rate to approximately 2 %. These winners all increased the number of layers used in the CNN, visualized in Figure 31. For comparison, the human rate in the same benchmark is about 5 %. [97]

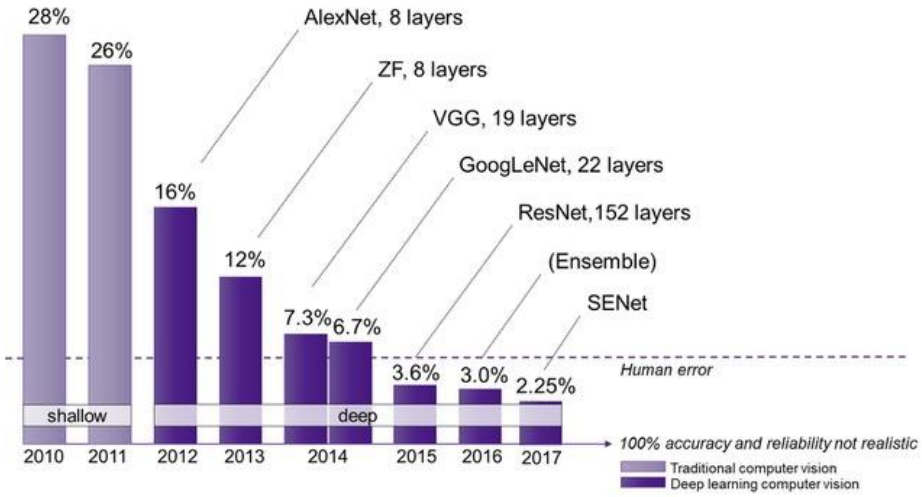


Figure 31: Accuracy of the winning systems since 2010 [98]

As described before, there have already been several object detection network developed. They are now good enough to be deployed in consumer products (e.g., Google Photos) and some are fast enough to be run on mobile devices. However, there are only a few papers of comparison of these convolutional object detectors. Mainly because the technology evolves so fast that any comparison becomes obsolete quickly [99]. This chapter should give a short overview of the speed and accuracy comparison of some convolutional neural networks [100]. A convolutional object detection system can be divided into two main categories: meta-architecture and feature extractor. Meta-architectures are: R-CNN, Faster R-CNN, R-FCN, Multibox, SSD and YOLO. Feature extractor are: VGG, MobileNet, Inception and ResNet. To describe individually how each CNN works would exceed this thesis. Hence, only an overview of the different performance is shown in Figure 32.

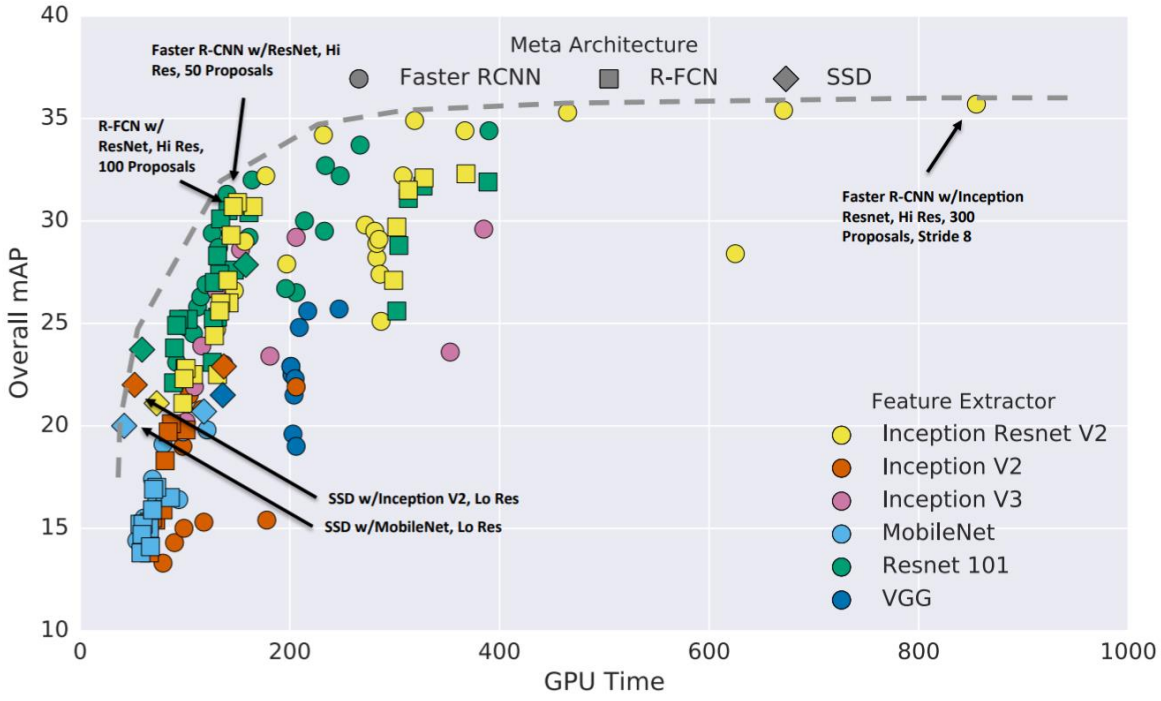
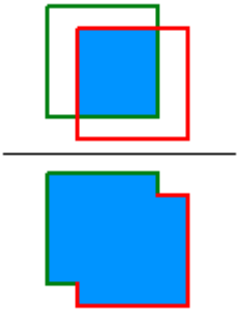


Figure 32: Comparison of different convolutional detection systems [100]

The Overall mAP is plotted on the y-axis. mAP stands for mean average precision. It is defined by the official COCO API, which measures mAP averaged over IOU thresholds in [0.5 : 0.05 : 0.95]. IOU is the Intersection over Union. [101]

$$IOU = \frac{area\ of\ overlap}{area\ of\ union} =$$



True Positive (TP) = A correct detection. Detection with IOU ≥ threshold
False Positive (FP) = A wrong detection. Detection with IOU < threshold

Precision is the ability of a model to identify only the relevant objects. It is the percentage of correct positive predictions and is defined as:

$$Precisicion = \frac{TP}{TP + FN} = \frac{TP}{all\ detetction}$$

Recall is the ability of a model to find all the relevant cases (all ground truth bounding boxes). It is the percentage of true positive detected among all relevant ground truths and is given by:

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{all\ ground\ truths}$$

The mAP is the precision averaged across all recall values between 0 and 1 and is defined as:

$$mAP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} \rho_{interp}(r)$$

With

$$\rho_{interp}(r_{n+1}) = \max \rho(\tilde{r})$$

Where $\rho(\tilde{r})$ is the measured precision at recall $\tilde{r}$.
A more detailed description of the calculation can be found in [100–102]

Table 7: Summary of CNN tested in [100]

Model	**mAP**	$\frac{\boldsymbol{GPU\ time}}{\boldsymbol{Accuracy}}$
SSD (MobileNet)	19.3	Fastest
SSD (Inception V2)	22	Fastest
Faster R-CNN (Resnet 101)	32	Sweet Spot
R-FCN (Resnet 101)	30.4	Sweet spot
Faster Inception (Resnet V2)	35.5	Most accurate

According to J. Huang, it can be difficult for practitioners to decide what architecture is best suited to their application [100],. Standard accuracy metrics, such as mean average precision (mAP), does not describe the whole requirements for a system, since for real deployments of Computer Vision systems, running time and memory usage are also critical [100].

These are additional critical requirements to consider:

- Training system (CPU or GPU based)
- Training time for CPU/GPU
- Possible Frames per Second of CNN (image or video)
- System Performance needed (Mobile phone, PC, etc.)
- Labeling process
- Number of labeled images
- Desired accuracy

For this thesis, the Faster R-CNN Inception V2 was chosen because it already performed well in a similar use case [103]. The concept of Faster R-CNN will be described in chapter 5.7.6.

4.4 Software development

The VDI 2206 serves as the basis for the further development of the Space Factory 4.0. It provides a useful frame for the design of any kind of mechatronic system. It consists essentially of three elements. The V model on the macro-level, as shown in Figure 33. A general problem-solving cycle on the micro-level and predefined process modules for handling recurrent working steps in the development of mechatronic systems.

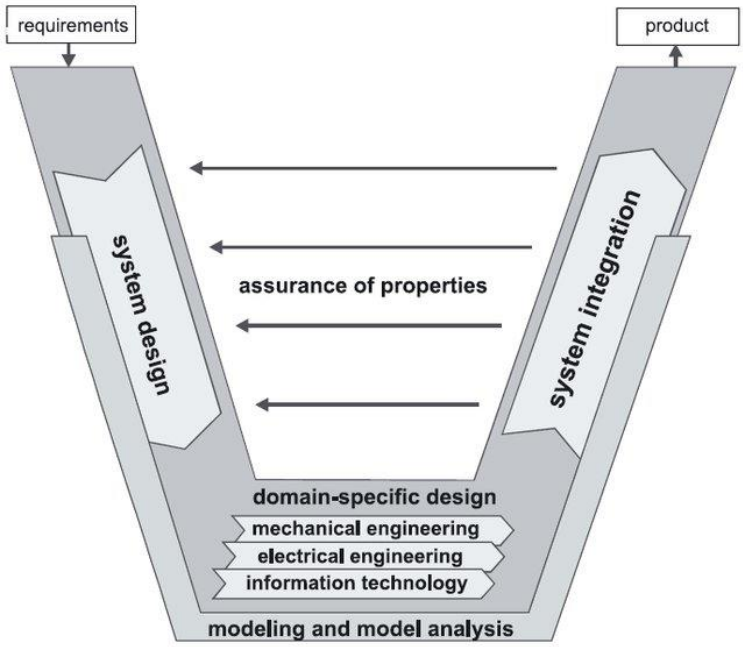


Figure 33: V model as a macro-cycle [104]

The V model describes the development process from the requirements to the finished product. The respective phases of the V model run iteratively. However, the individual phases are run through several times during the development process.
The starting point of the V model is the product requirements derived from the development order. The requirements describe the task precisely and clearly. In the following chapters, each table of requirements will be divided into functions that have to be implemented (“Must”) or can be implemented (“Can”). These requirements, at the same time, form the measure against which the later product is to be evaluated. In the subsequent system design, the system architecture is designed and a formal modeling of the overall functions is performed. The overall functions are broken down into main sub-functions for which suitable partial solutions must be found. The domain-specific design is the concretization of the previously developed partial functions with regard to the respective domain (mechanical engineering, electrical engineering, information technology). The respective solution elements are checked for function fulfillment in the entire system context with the help of detailed designs and calculations and are worked out independently of each other. Simultaneously, in the phase of modeling and model analysis, the system properties are mapped and examined by means of models and computer-aided simulation tools. Subsequently, the results of the individual domains are integrated into the overall system in the system integration phase. This phase aims to ensure that the results of the domains that are combined in the finished product work together without errors. Property validation is a continuous process of the entire development process. The progress is continuously verified against the requirements. The final result of a continuous macro-cycle is the product. A complex product is generally not produced within one macro-cycle. A product commonly repeats the cycle several times. [104]

Additionally, to this development process a different approach is known for machine learning development and data mining. This open standard is called the Cross-industry standard process for data mining (CRISP-DM). It was developed in 1996 with the cooperation of numerous well-known companies and consists of six different process phases. CRISP-DM is application-neutral and can be adapted to other areas. IBM adopted this standard and visualized the process, as shown in Figure 34. [105]

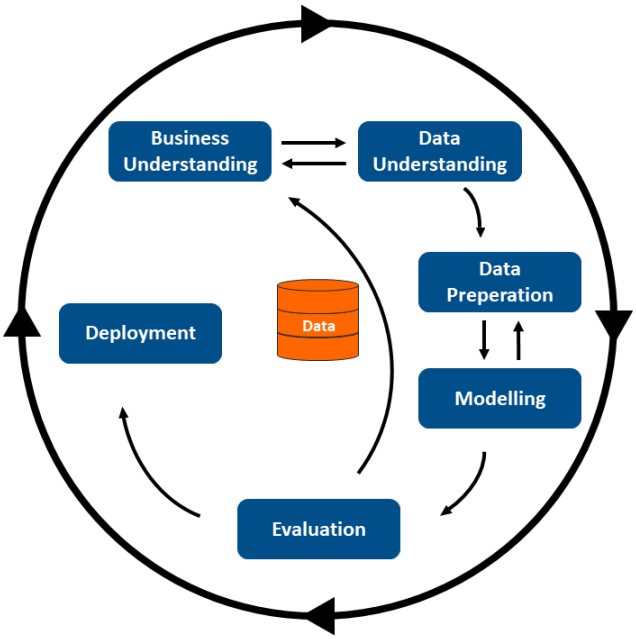


Figure 34: The data mining life cycle (based on [105])

CRISP-DM defines a total of six individual phases that must be passed through in a data mining project. However, it is not a one-time, linear run-through, because the individual phases can be repeated several times or it is necessary to switch between different phases several times. Depending on the results delivered by the individual phases, it may be necessary to go back to an earlier phase or to run through the same phase again. The six process phases are: [105]

1. Business Understanding

   In the business understanding phase, it is important to define the concrete goals and requirements for data mining. The result of this phase is the formulation of the task and the description of the planned rough procedure.

2. Data Understanding

   Within the framework of data understanding, an attempt is made to obtain an initial overview of the available data and their quality. The goal is to discover first insights or to detect interesting subsets to form hypotheses for hidden information.

3. Data preparation

   Data preparation covers all activities to construct the final dataset from the raw data. It is the most time-consuming part and devoting more energy in the previous chapter can be minimize this effort.

4. Modelling

   Within the scope of modelling, the data mining methods suitable for the task are applied to the data set created in data preparation. Typical for this phase are the optimization of parameters and the creation of several models.

5. Evaluation

   In this step, the results are evaluated using the business success criteria established at the beginning of the project. The evaluation ensures an exact comparison of the created data models with the task at hand and selects the most suitable model.

6. Deployment

   The final phase of the CRISP-DM is the deployment. In this phase, the results obtained are processed in order to present them and feed them into the decision-making process of the client. A final report is produced and a project review conducted.

As described in Crisp-DM and previous projects, the available data is at the center of any machine learning development process. Besides the quantity of data, the data quality is important and above all, the appropriate context of the data. Therefore before the actual learning phase, intensive human preparatory work is necessary to select good data from available project-owned databases. The quality of the AI system is dependent on the quality of the training data. [5, 105]

5 Concept Space Factory 4.0 with AI

In this chapter, the Space Factory 4.0 concept will be described in more detail. First of all, the Centralized Information Base will be clarified and expanded with artificial intelligence functions. A small concept of Predictive Maintenance is implemented and explained in chapter 5.5. The idea of a Self-Learning Production System will be introduced in chapter 5.6 and discussed theoretically. The main focus in this chapter is on the implementation of a Computer Vision system in chapter 5.7. A test set up for object detection, machine vision and synchronization with the digital twin will be showcased.

5.1 Space factory architecture

The concept of the Centralized Information Base (CIB) is described in “Space Factory 4.0 – New Processes for the robotic assembly of modular satellites on an in-orbit platform based on ‘Industrie 4.0’ approach” [26]. It represents a core function of the Space Factory 4.0 process. It is the database of definition and representation of reference models and instantiation of digital twins. CIB establishes continuous information exchange between digital twins, assembly support and control systems. It is an essential framework for integrating additional partial models through interfaces as modules to represent the whole life cycle of a CubeSat and information exchange. The partial models are shown in Figure 35 and consist of production, software, support systems & operation and product.

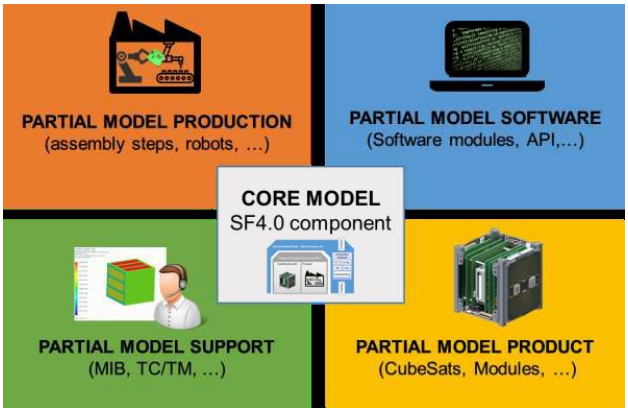


Figure 35: The concept of the Centralized Information Base

This Centralized Information Base is essential for the integration of artificial intelligence in the Space Factory 4.0 process. The conceptual idea of the CIB is to store all information or data of the Space Factory 4.0. This data can be used to build machine learning algorithms and implement the following previously described use cases: Predictive Maintenance (chapter 4.1), Self-Learning Production System (chapter 4.2), Computer Vision (chapter 4.3). To understand the idea of the CIB, it is transferred to already state-of-the-art systems used in Industry 4.0 production. Nowadays, a production facility uses different kind of application software in the whole process. Common application software are: Product Lifecycle Management (PLM), Enterprise Resource Planning (ERP) and Manufacturing Execution System (MES).

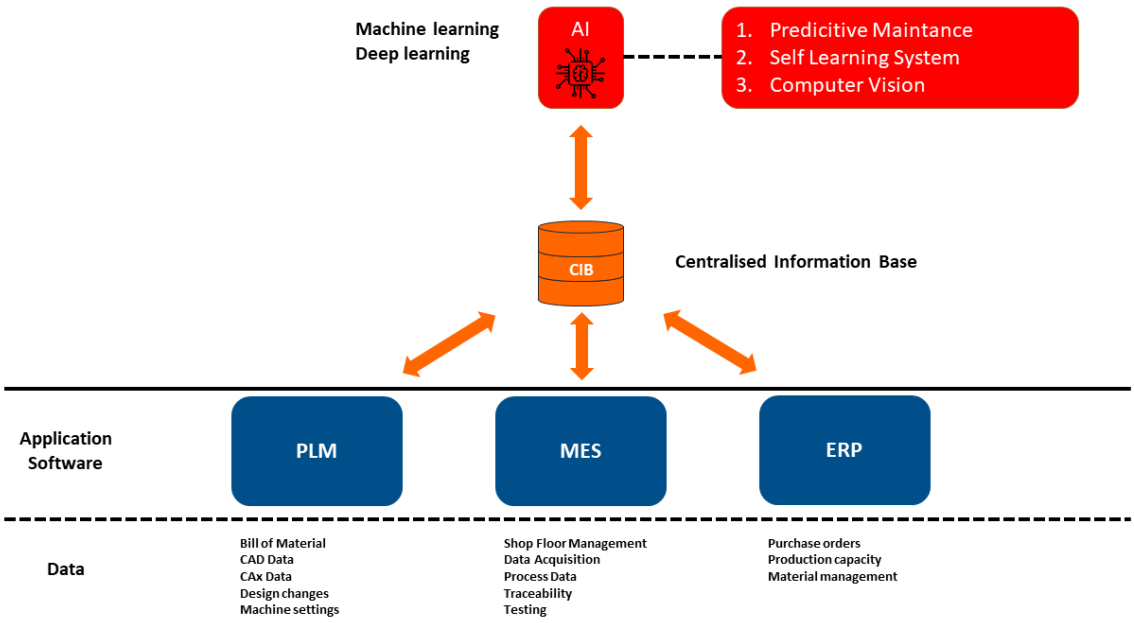


Figure 36: Space Factory process enhanced with AI

The concept idea is a collective term for all IT systems that enable central data storage of master and reference data and helps to improve data quality by ensuring that identifiers and other key data elements about those entities are accurate and consistent enterprise-wide. CIB provides data integration of the relevant process relations in development, production and assembly. Usually, Enterprise Resource Planning (ERP) is referred to as a business management software [106]. It stores business resources, such as purchase orders, production capacity and raw materials. Product Lifecycle Management (PLM) is not a defined software solution. Its goal is the management of the entire lifecycle of a product from concept to end of life [107]. Integrated are software tools such as: Computer-Aided Design (CAD), Computer-Aided Manufacturing (CAM) and Product Data Management. Relevant data are, for example: Bill of Material, CAx data, design changes, machine settings. Manufacturing Execution Systems (MES) are IT systems used in manufacturing. It generates current and even historical maps for production equipment and can thus be used as a basis for optimization processes [108]. It is mainly used for shop floor management and data acquisition of the machines. The MES collects data about product genealogy, performance, traceability, material management and work in progress.

Artificial Intelligence is not yet placed in the RAMI 4.0 architecture [5]. Figure 36 is a visualization of AI in the Space Factory 4.0 architecture and uses the same terminology introduced in RAMI 4.0. Data from the whole lifecycle of a product is necessary (“Life Cycle & Value Stream”) and is represented by the PLM system. Data from different hierarchy levels are also necessary. Data from the RAMI 4.0 layer “Work Centers” is represented by the MES system and the layer “Enterprise” by the ERP system.

5.2 General Requirements for Space Factory 4.0

The Centralized Information Base is an established system presented in previous works on Space Factory 4.0 [4, 26]. Besides the data exchange from PLM, MES and ERP the Space Factory 4.0 has to exchange data from the digital twin and its physical component. To visualize this cyber-physical system, the architecture from RAMI 4.0 is used in Figure 37. It is based on an asset (grey) and an administration shell (blue). The administration shell surrounds the asset

and provides it with compliant communication capabilities. On one side, the asset contains the physical component, such as the robot and its sensors and the CubeSat. On the other side, its corresponding digital twin, such as a CAD model or other forms of simulation. The Centralized Information Base is the main collecting point for this digital or physical generated data. To establish an artificial intelligence application, this database is used for training the algorithm.

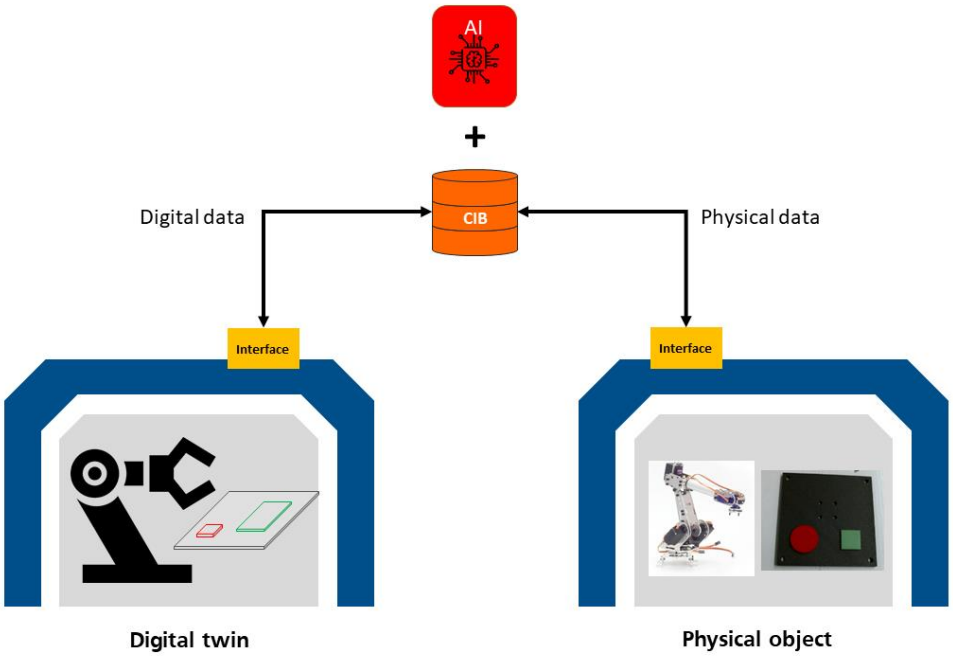


Figure 37: Cyber-physical system with CIB

This thesis is focusing on the previously mentioned AI applications: Predictive Maintenance, Self-learning Production System and Computer Vision. To integrate these in Space Factory 4.0, the following general requirements are made and expanded in the following chapters. The requirements are further subdivided into the priorities of the system. These help to understand which functions are essential ("Must") for the determination of goals. Functions that can be included or extend the system are marked with ("Can").

Table 8: General Requirements for Space Factory 4.0

Priority	Description
Must	**Communication between subsystems** A digital twin system and physical objects exist. The information must be transferred between these assets and synchronized. Exchange and management of data across application has to be possible.
Must/Can	**The system has to collect data** For real-world information the system has to be able to collect data from its environment. The data can be provided from IoT devices. The data can also be from a digital twin. The data has to be saved in the Centralized Information Base.
Must	**Digital twin entity unique** One physical object has to be clearly and uniquely assigned to one digital twin. So the digital twin ID is identifiable in the database.

Can	**The software should be consistent** For consistency the number of different programming languages should be kept to a minimum.
Must	**Integration AI** The CIB has to be accessible for the AI applications. The data has to be stored in a format that can be used for training the algorithms.
Must	**Autonomous** The Space Factory 4.0 is not always accessible in orbit. It has to run autonomous in several occasion. Through AI applications the Space Factory 4.0 reaches higher autonomy. At least a autonomy level of 3 has to be possible.

5.3 Experimental setup

For developing and implementing new applications in the Space Factory 4.0, an experimental test setup has been developed at the DIK. The physical part of the system is made up of a 6-DOF robot arm from Micropede [109]. This robotic arm contains six servo motors responsible for the rotation of its segments. All motors are controlled with a voltage of 6.0 V, a period of 19.5 µs and have a pulse width of maximum 500 to 2500 µs. The first four motors are DOMAN S2000MD servos, which have a maximum torque of 21.5 kg/cm. They are used for the heavier components of the robot, comprising a range of +-90 degrees. There are two servo motors on either side of the joint ate the base to compensate for the heavy weight. The base rotation is produced by an S2003MD, which has a larger range of 270°. For the finer movements of the end axes, two DOMAN S0090MD servo motors are mounted. They can only produce maximum torques of 2 kg/cm. The robot is roughly 180x130x230 mm and weighs 860 g. A gripper is also attached to the end of the robot weighing in at 110g with approximate measurements of 104x99x51 mm. It uses an S2000MD servo motor for opening and closing. It can grasp at a maximum width of 50mm. The exact dimension of the robotic is shown in Figure 38.

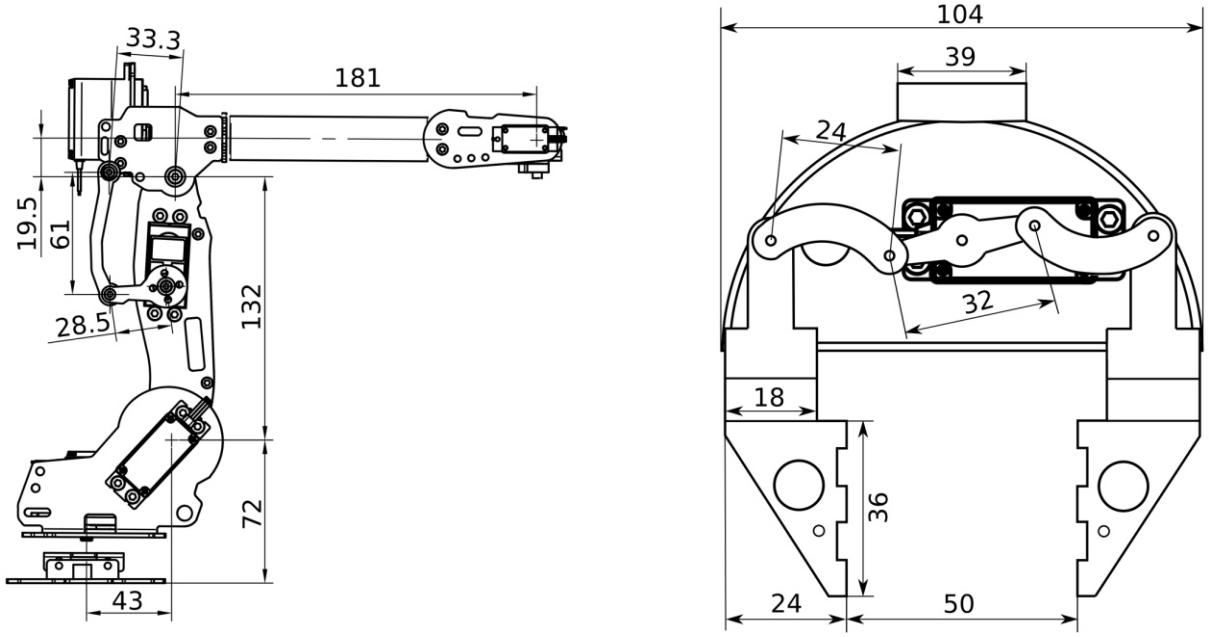


Figure 38: Robotic arm from Micropede (dimensions in mm) [109]

The current experimental setup of the Space Factory 4.0 concept consists of a low budget robotic arm, which is driven by servos and controlled by Tinkerforge modules [110]. These can consist

of different modules for communication, servo control and power management. They all can be stacked up together in a compact manner and easily extended with new modules. The robotic arm is also controllable with a VR interface. For this thesis, additional image processing equipment was added. The webcam C615 from Logitech with a resolution of 1080p is mounted onto the frame at the height of approx. 20 cm and pointed down to the placement platform. The frame of the Space Factory 4.0 is built with 1 cm wide black aluminum profiles mounted on top of a wooden board. A white placement platform serves for positioning the desired objects (e.g., CubeSat module). Positioned there, it is in the field of view of the webcam and tangible for the robotic arm. The robotic arm is limited in weight lifting. Fixing the webcam on the robotic arm would overload it as well as the servos. It would also result in unnecessary oscillations because then the robotic arm would be too top-heavy. Furthermore, the robotic arm is limited in accuracy of movement. The described servos have an accuracy of 2° or worse. With an arm length of 30 cm the accuracy is around 1 cm. It can perform simple pick and place operations. The whole set up is shown in Figure 39.

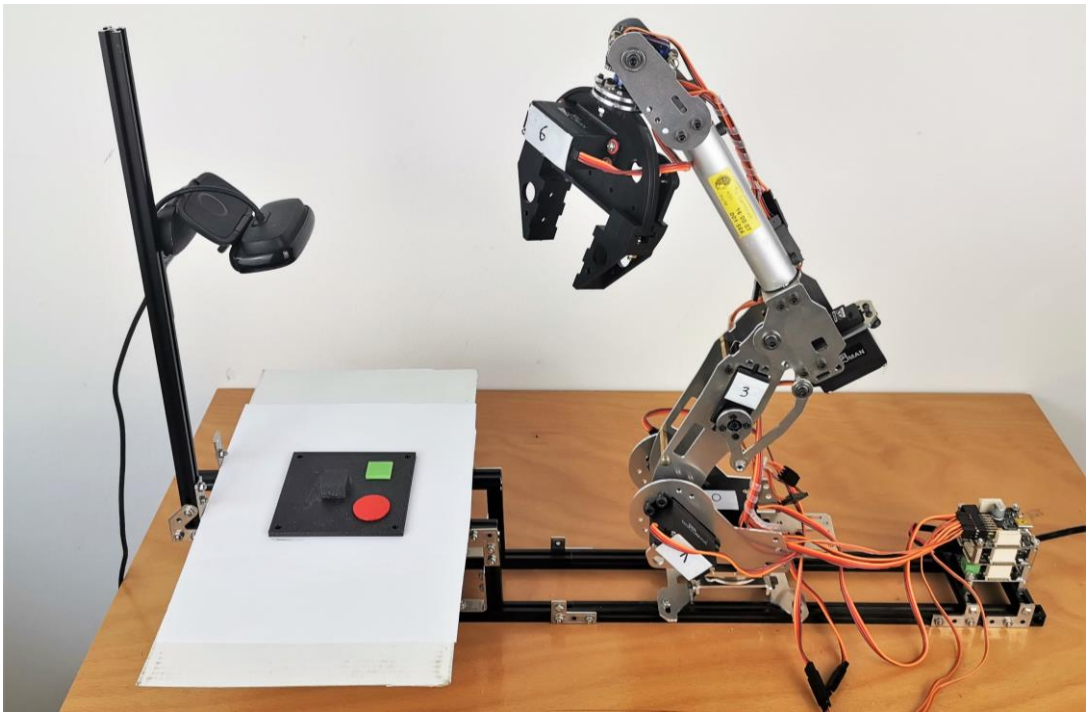


Figure 39: Experimental set up of Space Factory 4.0

5.4 Development environment

All program tools are developed on the Windows operating system, but can be executed on all systems that support Python. Python is an interpreted, high-level programming language with dynamic semantics and object oriented program paradigms are supported. Its simple syntax emphasizes readability and therefore reduces the cost of program maintenance. One key feature of Python is that it supports a variety of third-party modules and packages. This great open source library ecosystem is one of the main reason Python is the most popular programming language used for machine learning and data science [111]. To name some packages and libraries used in this thesis: TensorFlow, NumPy, scikit-learn, matplotlib and OpenCV. TensorFlow, developed by Google, is a common software library for machine learning and neural networks. It consists of pre-trained neural network models to choose from. NumPy adds support for large, multi-dimensional array and matrices and also a large collection of

mathematical functions. These are necessary for several operations in TensorFlow and OpenCV. OpenCV is a Computer Vision library [112]. This library has more than 2500 algorithms for classic Computer Vision and machine learning. For example, recognize faces, track objects and color detection. For a better graphical interface and debugging the integrated development environment (IDE) PyCharm is used. It contains a code editor, compiler and debugger, accessed through a single graphical user interface. For the creation of CAD models, the software Siemens NX is chosen, which is a common tool in several projects at DIK. The digital twin of the CubeSat is modeled in Siemens NX. NX has the ability to run functions in different programming languages that are created through an API called NX Open. Unfortunately, the function for Python is limited and a real-time server-client connection between the Python tool and digital twin is not possible. Therefore, a text file is shared between the two programs for data exchange. This text file can be executed in NX Open, which is called NX Journal. To maintain consistency, the TinkerForge modules are also programmed in Python for access to the robot arm's servo motors and compatibility with the other subsystems. An overview of all relevant software tools or libraries are shown in Figure 40.

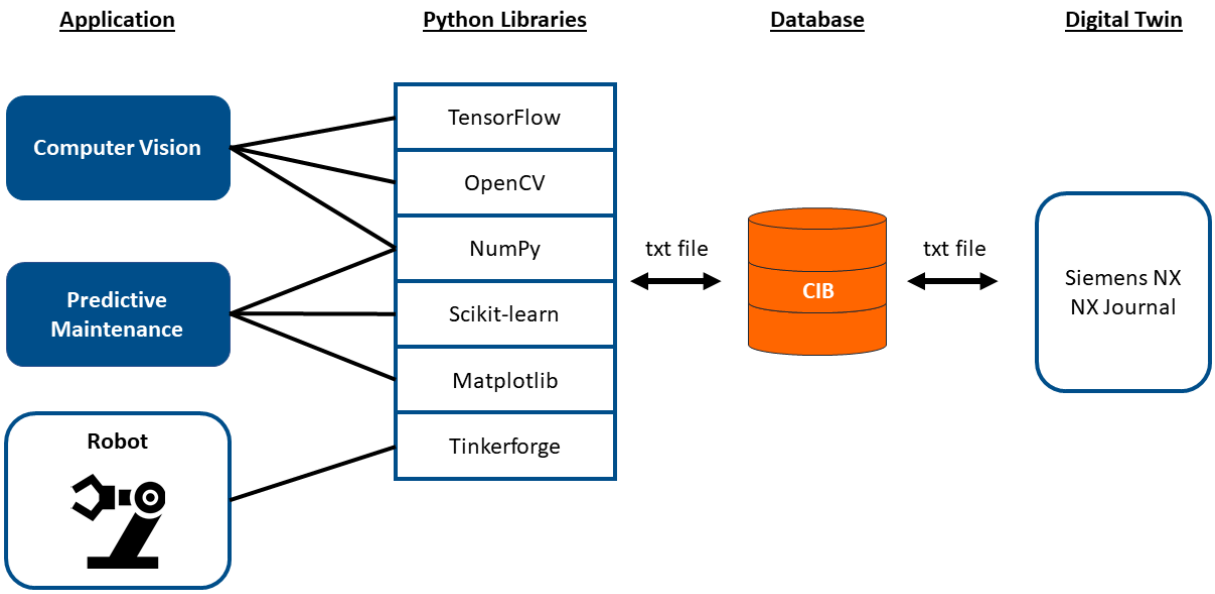


Figure 40: Overview of relevant software tools

5.5 Predictive Maintenance in Space Factory 4.0

The fundamentals of Predictive Maintenance have been elaborated in the previous chapter. In this chapter, a concept is developed to implement Predictive Maintenance in the Space Factory 4.0 process. A pilot program is tested to prove the functions of data collecting, data preparation and the use of machine learning algorithms. The classification algorithm used in this case is called Support Vector Machine.

5.5.1 Concept Predictive Maintenance

Collecting Data is the key component of any Predictive Maintenance program. Therefore, there must be technology in place to collect, process, prepare and structure massive amounts of device data which will be stored in a database. This system must be able to understand what each piece of data represents so that it can be monitored and labeled. In the work frame of Space Factory 4.0 this is the Centralized Information Base. The concept of implementing Predictive Maintenance depends on the data stored in the CIB, as visualized in Figure 41. All data has to be stored in this centralized database to be further analyzed. The data can be provided by physical components which have built-in sensors to monitor the condition of the components such as temperature or vibration. Simulation data can also be provided by a digital twin. If a simulation of one health condition is possible with mathematical definition, it can be integrated in the value chain. All this data integrated in the CIB is further analyses with machine learning algorithms as described in chapter 4.1 and different Predictive Maintenance platforms are described in chapter 4.1.5.

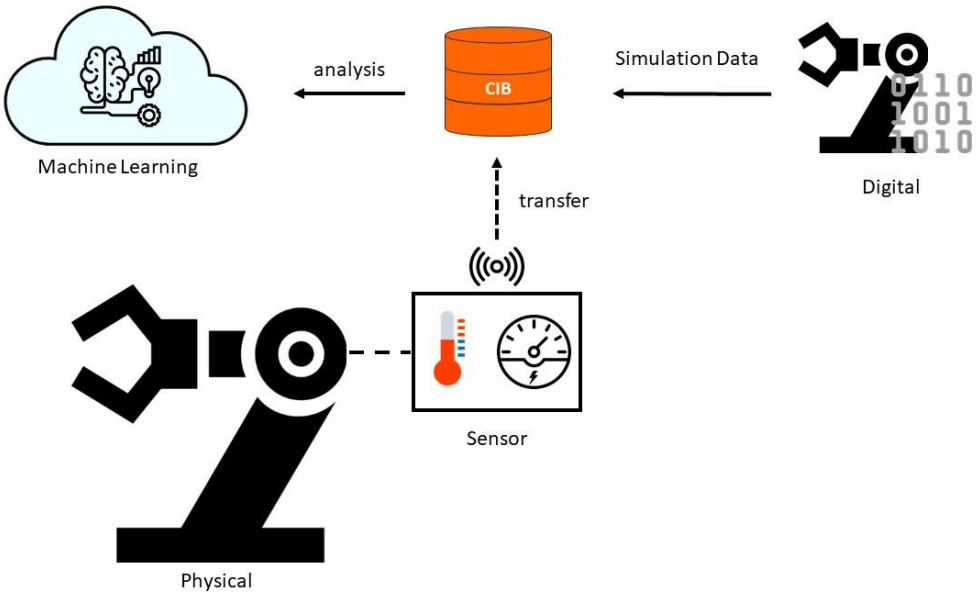


Figure 41: Concept for Predictive Maintenance

One component that can be exploited is the servo motors of the robotic arm. The servo motor is a critical component because it is necessary for the whole movement of the robotic arm. The major difficulty for implementing Predictive Maintenance is collecting useful data and also labeling of this data. To develop an algorithm, there needs to be enough labeled data of the health condition of the machine. Therefore, the next chapter describes how Predictive Maintenance can be in the Space Factory 4.0 set up.

5.5.2 Requirements

The requirements describe which functions are necessary for implementing Predictive Maintenance in Space Factory 4.0. The requirements are further subdivided into the priorities of the system. These help to understand which functions are essential ("Must") for the determination of goals. Functions that can be included or extend the system are marked with ("Can").

Table 9: Requirements for Predictive Maintenance

Priority	Description
Must	**Sensor data** The system has to measure data from its machines and devices. (e.g., servo) The data can be voltage, current, force, etc. Additional sensors should be extendable. The data must be transferable to the Centralized Information Base.
Must	**Failure history** For building prediction models, the algorithm has to learn about a components normal operational pattern, as well as its failure pattern. The training data should contain a sufficient number of examples from both patterns. Data can be from maintenance records and parts replacement history. (e.g. servo, bearing)
Can	**Time steps of data** The time interval for measuring the data has to be defined. For example, every 10 minutes. This interval can vary if the collected data should be analyzed over a time period of a day, days or even months.
Must	**Machine learning algorithms** Different machine learning algorithms have to be applicable. For example, Regression models are used to compute the remaining useful life (RUL).
Must	**Centralized Information Base** The Centralized Information Base has to be integrated. The measured data has to be transferred to the CIB.
Can	**Digital twin** A simulation/digital twin of the machines behaviour can also be integrated. In this case, the simulation data should be transferred to the CIB for further processing.

5.5.3 Pilot program

Before implementing Predictive Maintenance in a whole factory, it is useful to start with a small pilot program. One critical component in the Space Factory 4.0 is the servo motor of the robotic arm. If one servo motor is failing, the whole robot arm is compromised. Therefore, the following experiment will be conducted on the servo motor. The general steps for implementing Predictive Maintenance in Space Factory 4.0 would be:

1. Establish a strategy
2. Choose the right asset and algorithm to test
3. Develop a pilot program to prove the concept

Future goals can be:

4. Build a data mining strategy
5. Scale to more assets and different algorithm

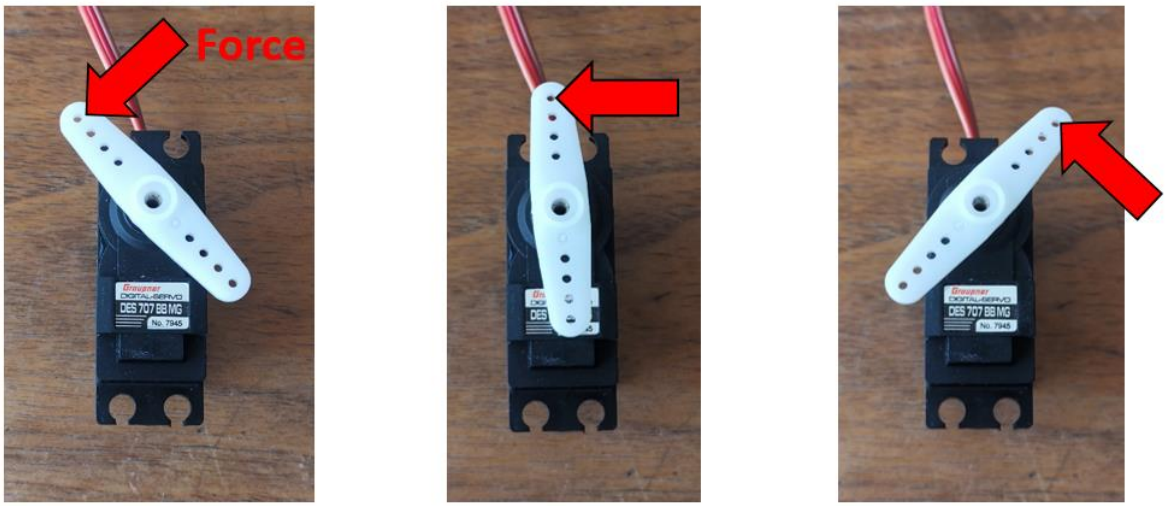


Figure 42: Test set up for servo motor

5.5.4 Data collecting and analysis

The first step, as described in chapter 4.1.1, is to collect and analyze the data of the servo motor. The servo to be tested is called Graupner DES 707 BB MG, as shown in Figure 42. It is connected to the TinkerForge servo module. In the first step, all available data will be measured so machine learning algorithms can be applied. The TinkerForge API allows it to control the servo with Python commands. The servo parameters are set to: Voltage = 6 V, pulse width = 1000 – 2000 μs, period = 20000 μs, acceleration = 100 $\frac{^\circ}{s^2}$, velocity = 200 $\frac{^\circ}{s^2}$. . The servo is moved to the position -45° and 45° in a loop, as shown in Figure 42. While the servo is moving the following data can be measured every 0.05 s in a timeframe of 10 s. The terms in the brackets are the corresponding python functions from the TinkerForge module. The position (servo.get_current_position()) and velocity (servo.get_current_velocity()) is visualized in Figure 43.

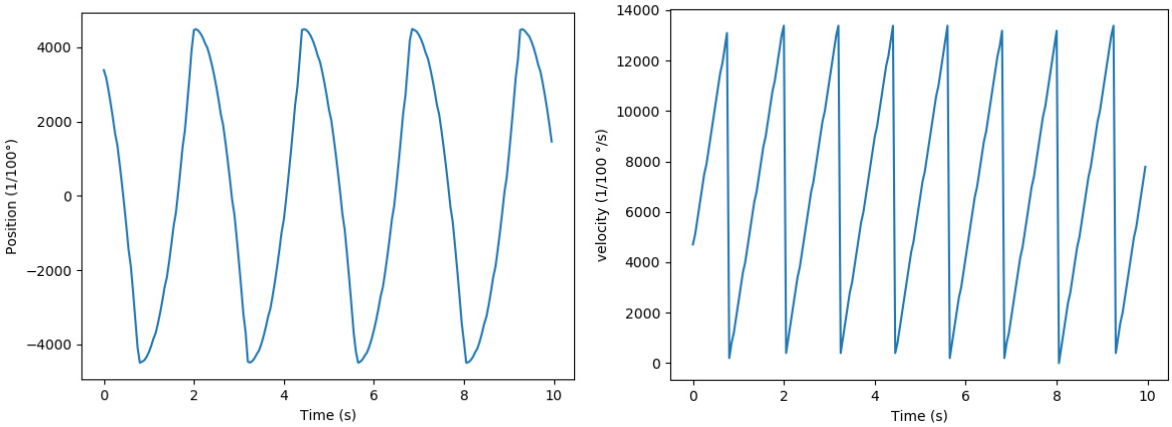


Figure 43: Position and velocity graph

The current/amperage consumption (servo.get_servo_current()) and the temperature in the chip of the TinkerForge servo brick (servo.get_chip_temperature()) is shown in Figure 44.

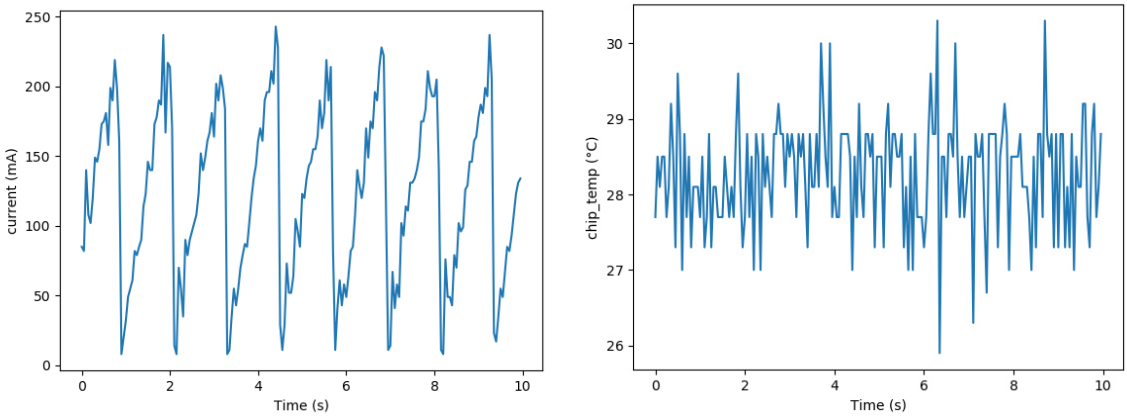


Figure 44: The current (left) and the temperature of the brick chip (right)

Position, velocity, current and temperature are the only real-time available data build inside the components. Hence, no additional sensor data is used; only this data can be applied to a machine learning application.

5.5.5 Data Preparation

Since the available data is limited, only a simple classification analysis can be conducted. First, the servo is moved in a loop from position 45° to -45° with no additional force applied. This scenario is defined as the normal operation (OK). The abnormal operation (NOK) is when force will be applied to the servo in operation. In this experiment, the force will be applied manual by hand. The only parameter that can be used for this anomaly detection is the current consumption. Position, velocity and temperature did not change in the different operational states.

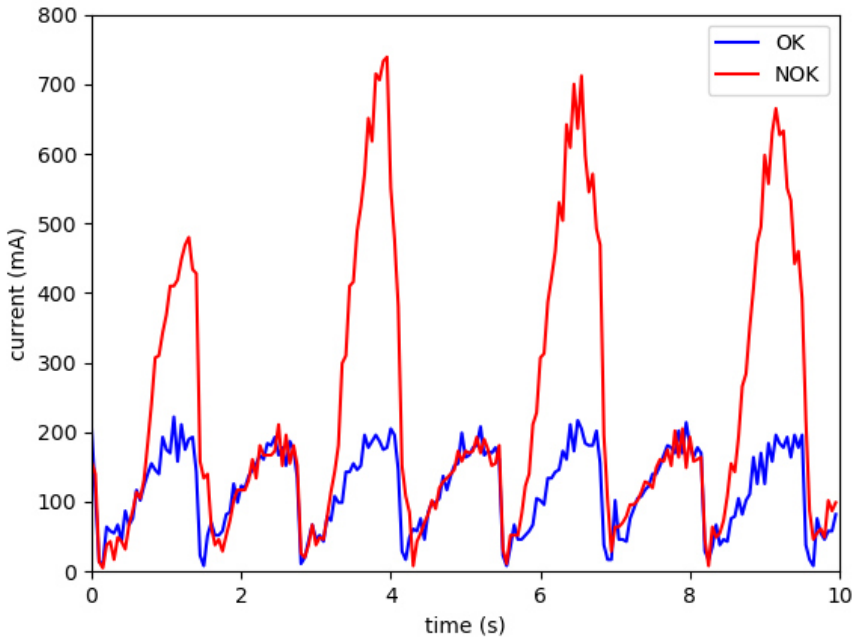


Figure 45: Comparison of current in OK and NOK scenario

An example of one measurement of a series of measurements is shown in Figure 45. The current of the normal state is shown in blue and labeled with OK. The current while applying force is shown in red and labeled with NOK. The current of the abnormal state is significantly higher at some time points than the current of the normal state. This correlation is analyzed in 20 more measurements and used for a classification algorithm.

5.5.6 Support Vector Machine

```
from sklearn import svm
from sklearn.model_selection import train_test_split
# labelled data OK=0; NOK=1
target = np.array([0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1])
# import servo measurements
data = np.load('data.npy')
# define support vector machine algorithm
classifier = svm.SVC()
# split the data in training data and test data randomly by 50%
X_train, X_test, y_train, y_test = train_test_split(data, target, test_size=0.5, shuffle=True)
# learn measurement on the first half of the measurement
classifier.fit(X_train, y_train)
# predict the value for second half of the measurement
prediction = classifier.predict(X_test)
# show the prediction
print('Prediction', prediction)
# label prediction with OK and NOK
prediction = np.where(prediction == 0, 'OK', prediction)
prediction = np.where(prediction == '1', 'NOK', prediction)
print(prediction)
# visualize the first 3 predictions
x_axis = np.arange(0,10, 0.05)
plt.xlabel('time (s)')
plt.ylabel('current (mA)')
plt.axis([0,10,0,800])
plt.plot(x_axis, X_test[0], 'b', label=prediction[0])
plt.plot(x_axis, X_test[1], 'g', label=prediction[1])
plt.plot(x_axis, X_test[2], 'r', label=prediction[2])
plt.axis([0,7,0,800])
plt.legend()
plt.show()
```

Figure 46: Extract of the anomaly detection code with SVM

For defining the machine learning algorithm, the library scikit-learn [71] is imported. An extract of the code for anomaly detection of the servo motor is shown in Figure 46. Additional libraries are NumPy [113] and Matplotlib [114]. First, the labeled data for the state OK and NOK (target) and the measurement data (‘data.npy’) are imported. Because it is a pilot program the measurement data is limited to 20 records. Theses records are labeled for the OK state with 0 and with the NOK state with 1. After the labeling process the classification algorithm is used. The standard Support Vector Classification (SVC) algorithm is based on libsvm mathematical definition [115]. A Radial Basis Function is used for the default kernel type. Additional parameters to define the SVM algorithm can be adjusted such as gamma, coef0 and degree of the polynomial kernel function. As other classifiers SVC takes as input two arrays: an array X of size [n_samples, n_features] holding the training samples, and an array y of class labels (strings or integers), size [n_samples]. The measurement data is randomized to avoid biased training. The function train_test_split() splits arrays into random train and test subsets. Half of the data is used for training and the other half for testing. Classifier.fit() fits the SVC model according to the given training data. Classifier.Predict() performs classification on samples in X_test. The prediction is then printed and labeled with the string values ‘OK’ and ‘NOK’. The first three values of X_test are plotted and shown in Figure 47.

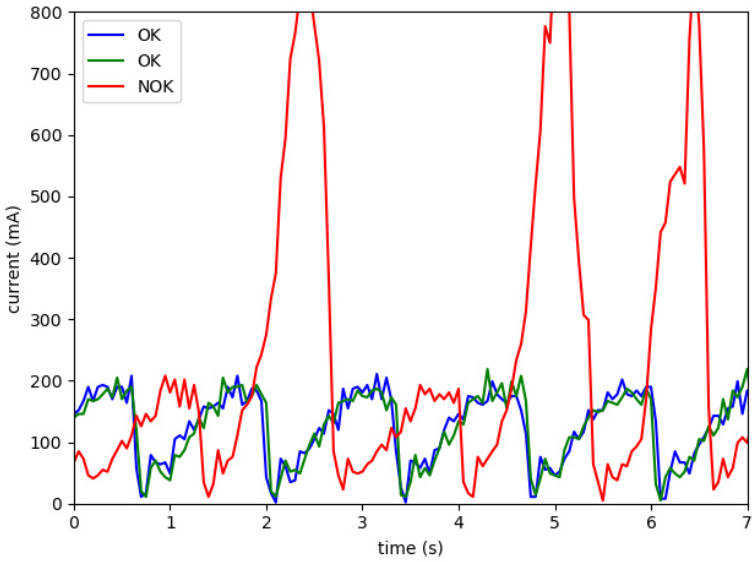


Figure 47: Visualization of three anomaly detection classifier predictions.

The prediction process is repeated several times with different data inputs and showed correct classification every time. Therefore, no parameter optimization of the SVC classifier is necessary. Figure 47 shows that the red plot is successfully classified as NOK. The blue and green plots are successfully classified as OK.

5.5.7 Conclusion

First of all, the small experiment showcases the successful implementation of a Support Vector Machine method for a simple classification problem. The classification problem is simple because only two classes as a target are defined: OK, NOK. A SVC also supports multi classification, which may not be relevant for anomaly detection. Because only the normal and abnormal state is of interest. Another factor is more relevant. Currently, the only used data is the current of the servo. The current is the only available and sufficient interpretable data and limits the experiment. The experiment can be expanded with additional sensors (e.g. vibration and torque) to generate more data to make the prediction more reliable. If a feature identification is possible with these new datasets, they need to weighted and normalized for further implementation. Furthermore, other classification states can be possible. Predicting the Remaining Useful Life (RUL) can also be a great field of interest. In this case, time-related data of several working hours or days is necessary. It is also difficult to find a data source that describes the health condition of the servo motor. Temperature and vibration can probably be a promising data source for describing the health condition, but further investigation is necessary. The digital twin can have a more relevant role in this Predictive Maintenance use case. When a useful data source is found for predicting RUL, for example vibration. The digital twin can be used to simulate hours or days of operating time and produce digital simulated sensor data. The simulated environment can iterate on much faster pace than the servo in the real world. So a test set up in the real world would cost days but could be simulated in hours with the digital twin. The only difficulty is that the failure scenario also needs to be simulated. For instance, if vibration is a promising indicator, the digital twin needs to be able to simulate wearing and deformation.

Implementing Predictive Maintenance applications in the Space Factory 4.0 is an iterative process. This pilot program showed the successful classification of one problem with one algorithm based on one dataset. More algorithms and datasets should be considered for solving even more complex problems.

5.6 Self-Learning Production System in Space Factor 4.0

The idea of a Self-Learning Production System integrated in the Space Factory 4.0 is only conceptual and the necessary requirements will be introduced in the following chapter. It is mainly based on reinforcement learning and the research presented in chapter 4.2.

5.6.1 Concept Self-Learning Production System

Also in the Space Factory 4.0, the use of reinforcement learning can be implemented. Currently the motion of the 6DoF robot arm is controlled traditionally. The robot arm can only move to predefined paths or it can be guided manually by an operator via VR controller. In the Space Factory 4.0 one scenario is picking and placing of different CubeSat modules. The picking motion can vary with different modules. Therefore the use cases mentioned the in chapter 4.2 can be transferred to the Space Factory 4.0 project. Different algorithms, such as Q-learning, SARSA, etc. can be further researched and compared. The experiment can be conducted in the following configuration. The CubeSat modules are placed in random configurations. In a real Space Factory 4.0 in orbit, the modules could also fly around because of zero gravity. First, the robot learns how to pick up a module and then uses this knowledge to pick it up in zero gravity. Secondly, the robot can analyze the environment because of object detection (described in 4.3.5) and decides how to pick up the object because of reinforcement learning. The scenario is visualized in Figure 48. The CubeSat modules are placed randomly in the environment.

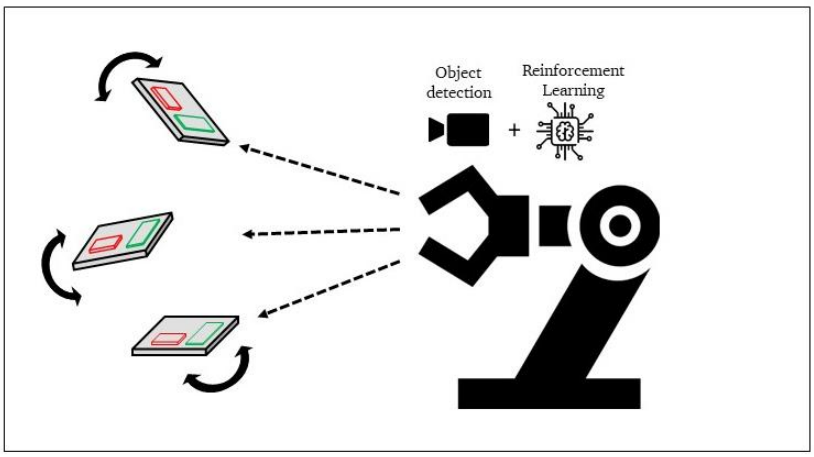


Figure 48: Space Factory 4.0 with reinforcement learning

5.6.2 Sequence

The system is built out of two subsystems. The object detection system and the control system of the robot with reinforcement learning ability. The object detection system can be structured as analyzed in chapter 5.7 and will not be further investigated in this chapter. After the object is successfully detected, the sequence is executed, as shown in Figure 49.

Subsequent to the object detection the system knows if the object has been picked up before. If no movement protocol is available, the control system has to learn how to pick up the part. This training situation can happen in the real world or with the digital twin. If the robot is used in the real world a controlled environment has to be provided. For learning the object needs to be analyzed 3-dimensional so the control system knows where the potential grabbing positions are. After the iterative process of learning with reinforcement learning, as described in 4.2, the

movement protocol has to be saved. It needs to contain information about the grabbing position and movement of the motors of the robot. Now the training is completed and the movement protocol can be used for the same object at a different time point. In this case, the sequence where the object is already trained can be executed ("YES"). The movement protocol and the grabbing positions are loaded. The object will be analyzed 3-dimensional to know how the object is orientated and to find the grabbing positions. In the end, the known movement is executed.

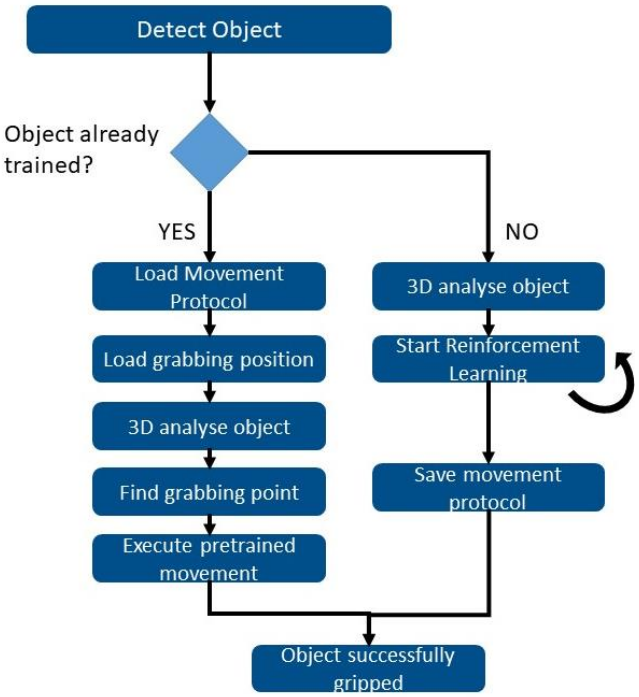


Figure 49: Sequence of robot execution with reinforcement learning

5.6.3 Requirements / Discussion

In this chapter it will be discussed how the self-learning system can be implemented in the Space Factory 4.0. It will be elaborated which systems of the current set up need to be changed.

Simulation of robotic arm

Currently, the digital representation of the robotic arm used in the Space Factory 4.0 test set up is visualized in the Siemens Mechatronic Concept Designer. A csv file is exchanged between the robot controller and Siemens NX. This set up cannot be used for reinforcement learning. The time delay and accuracy is too low and Siemens NX does not allow reinforcement simulations.

Simulation Platform

Reinforcement learning only operating on a physical component is time-consuming. It would also be dangerous to operate the robotic arm freely in the real world. Hence a simulation environment is recommended. Common software libraries for reinforcement learning are, for example, the GYM API from Open AI. [116] Gym is a toolkit for developing and comparing reinforcement learning algorithms.

Framework
The current experimental set up is not robust and uniformly. It was developed in different software tools and programming languages. With reinforcement learning additional libraries are necessary and the software framework will get more complex. Hence, a robotic middleware can be used, such as the Robotic Operating System (ROS). Such systems are designed to manage the complexity and heterogeneity of the hardware and applications.

Transferability
Research in reinforcement learning is often performed only on simulations. The algorithms are tested in the simulated environment. The Space Factory 4.0 system should also perform the simulated task in the physical world. Therefore, the trained knowledge needs to be transferable to the physical component.

Physical components
The current performance of the robotic arm is limited and shaky. It cannot perform the same tasks as common industrial robots that would be used in the real Space Factory 4.0. The robot version used in the test set up can be scaled down but needs to be stabilized and accuracy increased. The current servos have an accuracy of 2° or worse. With an arm length of 30 cm the accuracy is at the Gripper at around 1 cm. For complex tasks and grasp procedures, a higher accuracy is needed.

Object detection
For object detection an additional system is needed. It should be implemented on a convolutional neural network. It can be based on the system introduced in chapter 5.7. All relevant objects for grasping have to be trained. Additional objects that disorder (e.g., screws, debris) can be trained. So they can be maneuvered around.

5.6.4 Conclusion

All in all, reinforcement learning is a promising technology. The technology is still in its early phase. Only a little relevant research in the frame of Industry 4.0 was done so far and a few scientific papers are available. Reinforcement learning is a complex technology because the agent needs feedback from the environment. This environment has to be modeled in some way. Therefore, a small pilot program should be considered for starting reinforcement learning, instead of simulating the whole 6-DoF robotic arm. In the end, the whole Space Factory 4.0 could be optimized with reinforcement learning in a way that cannot be developed by a human. In addition, the Computer Vision system described in the next chapter can be used for further implementation and accelerate the development of a Self-Learning Production System in some areas such as object detection.

5.7 Computer Vision

In the following chapter the implementation of a Computer Vision enhanced system for the Space Factory 4.0 will be described. First of all, the requirements for the Computer Vision system will be determined. Afterwards the object detection and quality inspection will be clarified. In the end, the whole system will be evaluated and how it can be extended in the future.

5.7.1 Concept

To visualize the idea of a Computer Vision enhanced system for the Space Factory 4.0, Figure 50 is shown. The robotic arm is equipped with some kind of digital imaging device (e.g., webcam) to inspect the object it needs for the Space Factory 4.0 process, such as a CubeSat Module. The system should be able to detect the object with a Convolutional Neural Network and inspect other parts with machine vision. The measured values need to be synchronized with the CIB and the digital twin.

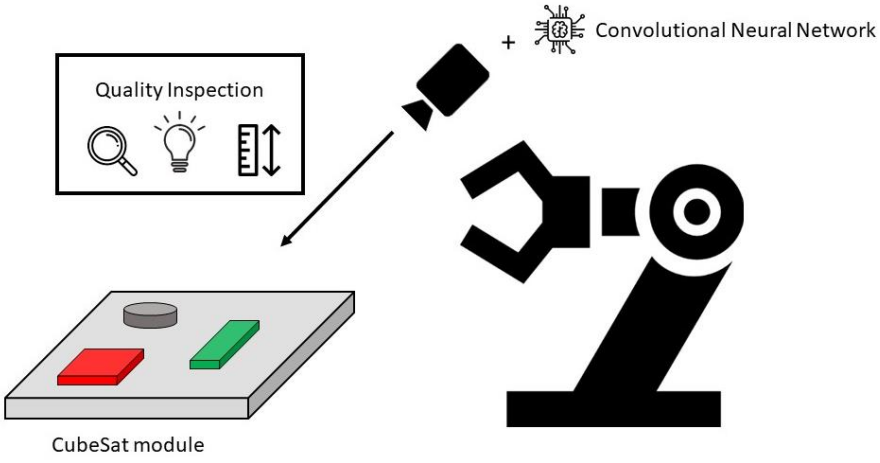


Figure 50: Space Factory 4.0 enhanced with Computer Vision

5.7.2 Requirements

The requirements describe which functions are necessary for implementing the Computer Vision system in Space Factory 4.0. The requirements are further subdivided into the priorities of the system. These priorities help to understand which functions are essential (“Must”) for the determination of goals. Functions that can be included or extend the system are marked with (“Can”).

Table 10: Requirements for Computer Vision

Priorities	Description
Must	**Object detection** The system has to detect automatically which kind of object is in its field of view. The number of objects detectable is defined in the Centralized Information Base. The number of detectable classes should be expendable. It must be able to segment multiple objects in the field of view.

Can	**Speed** The system should operate in real-time. That means it can operate in the frame rate like a human visual system (12 frames per second).
Must	**Gauging and measurement** The system must be able to measure specific elements in the field of view of the system. The measuring system can be optical in any form (e.g. laser, camera). The measured data has to be transferred to the Centralized Information Base.
Must	**Presence / Absence** The system has to detect if a desired part is present or absent. The information has to be transferred from the digital twin.
Must	**Inspection and defect detection** The system has to be able to conduct an optical quality inspection and detect if a defect on the part is present. If the part is defect, a warning has to occur and part has to be removed from the process. The definition and data of a defect has to be provided by the CIB.
Must	**Assembly verification** The system has to be able to check if the parts are assembled correctly. If the final assembly is correct the process is finished and the object can be deployed.
Can	**Robotic guidance** The system can also be used for guidance of the robotic arm. It can provide spatial information where the object is located and/or where the robotic arm is located.

5.7.3 Sequence

Figure 51 shows a flow chart of the simplified system. It only consists of pseudocode to describe in natural language what the major operations are. One operation is symbolized in a blue box and consists of multiple sub-functions which are not visualized. First, the webcam scans the environment. It used a Convolutional Neural Network to detect objects. Which is described in more detail in chapter 4.3.5 and 5.7.6. If the detection was successful, the quality inspection of the object starts. In this case, the object is a simplified CubeSat module. Then the system checks if the parts (red circle, green rectangle) of the object are present. It mainly uses color filters and counting pixel functions to determine the existence. Afterwards the information of the digital twin is imported from the Centralized Information Base. In this case, the information consists of dimensions and color numbers of the parts. But it can be easily extended with more information (e.g. tolerances, access points for the gripper). Then the measured values are visualized (shown in Figure 59) and the updated values are transformed into a text file and send to the Centralized Information Base. This information is used to generate an updated digital twin with new dimensions. The last step is the execution protocol of the robotic arm (shown in Figure 51). It moves to the desired position, opens the gripper, grasps the object and places it on the target position.

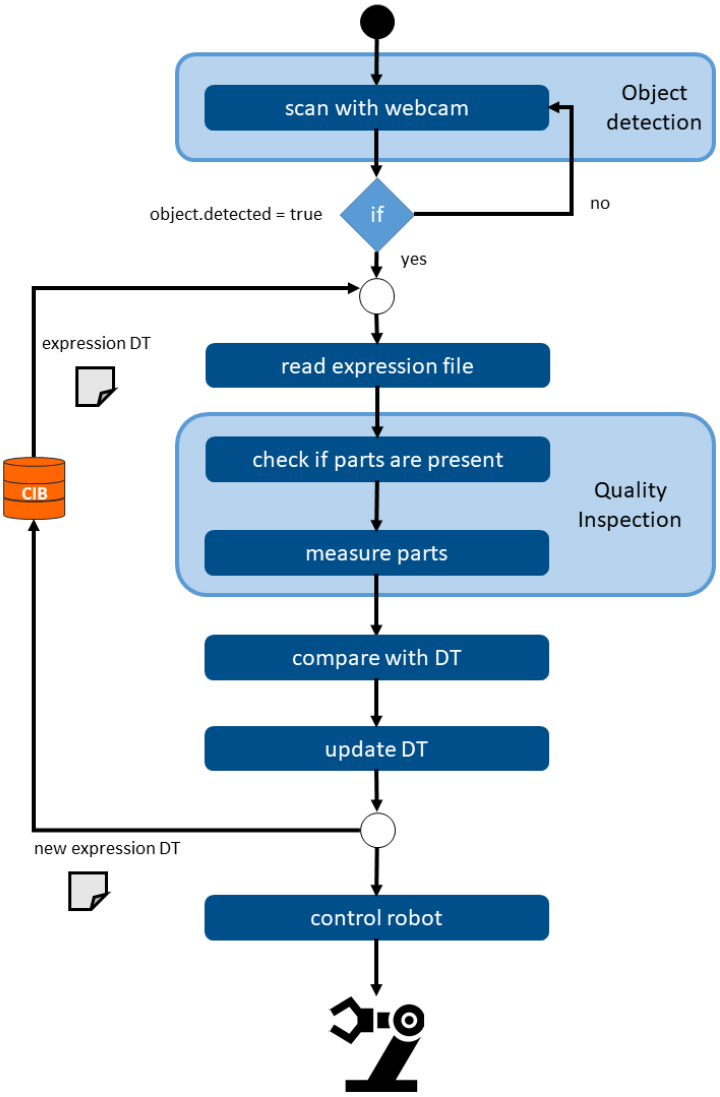


Figure 51: Flow chart of the system

5.7.4 Implementation

A simplified CubeSat module is created for object detection. Is has a size of 80 x 80 mm. And consist of different parts mounted on the plate. A red circle and a green rectangle, which are parts of the component group of the CubeSat module. As shown in Figure 52, a real CubeSat module consists mainly out of rectangular and circular parts.

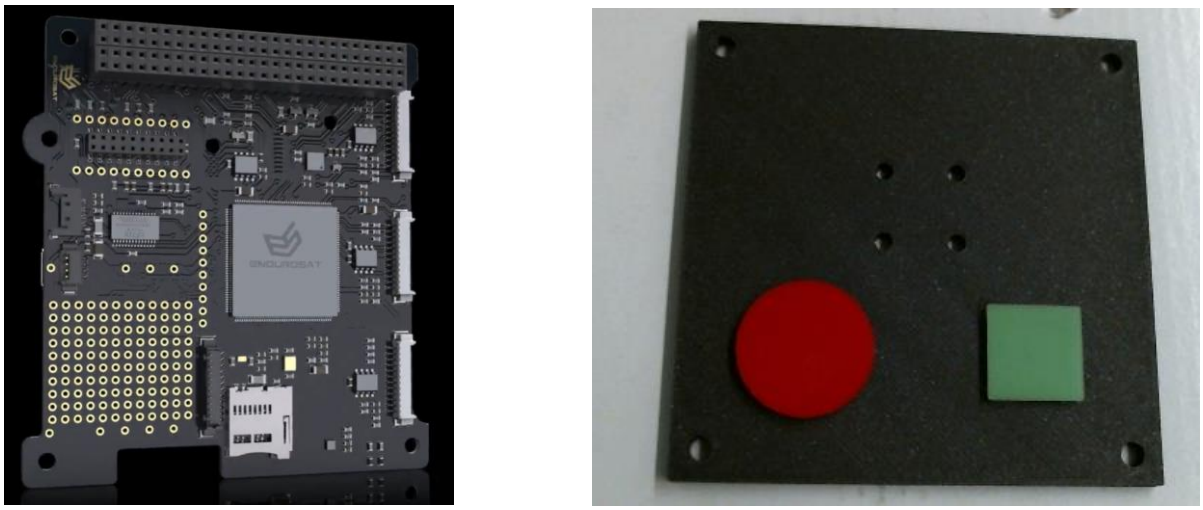


Figure 52: Comparison of a real CubeSat module and the abstract version [117]

A CAD model is created to represent the digital twin. Typically, a PLM system is used in combination with a CAD System to save product information. The PLM system developed for Siemens NX is called Teamcenter. For this thesis, this feature is not available. Hence, a different approach is used to simulate a simple form of a data exchange system. When designing a part, the feature “Expression” can be used to save values (e.g. dimensions). These values can also be used for inter part usage. For example, if one dimension is defined in a part and another part can get access to it. The data is shared from top to bottom. The following data is stored, as shown in Table 11.

Table 11: Information saved in “Expression“

Expression name	Value	Description
cubesat_length	80	Outer dimension of the CubeSat module. In millimeter.
cubesat_color_number	173	The color number for the CubeSat. Number defined by NX
circle_color_number	186	The color number for the circle. Number defined by NX
diameter	22	The diameter of the circle. In millimeter.
rectangle_length	15.5	The dimension of the rectangle. In millimeter.
rectangle_color_number	29	The color number for the rectangle. Number defined by NX
grab_pos_x	75	X-coordinate of the module where it should be gripped.
grab_pos_y	40	Y-coordinate of the module where it should be gripped

To export the files as a text file, a script is written in visual basic. It can be executed through NX journal. A short version of the file is shown in Figure 53. It opens a window in NX and writes all defined expressions into the window. Then the text in der window can be saved as a text file. This text file can also be integrated into the Centralized Information Base (described in chapter 5.1). Under “Select Case temp.Type” the different cases are defined what should be executed under which condition. For example, if the expression is a “Number” the name of the expression is written (lw.WriteLine(temp.equation)) and its corresponding value (lw.WriteLine(temp.equation)) to the NX command window.

```
Option Strict Off
Imports System
Imports NXOpen

Module Module1

    Sub Main()

        Dim theSession As Session = Session.GetSession()
        Dim workPart As Part = theSession.Parts.Work
        Dim lw As ListingWindow = theSession.ListingWindow
        Dim myDocs As String
        myDocs = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)
        Dim outputFile As String = "C:\Users\spacefactory\Desktop\Marcel\expressions.txt"

        'get information about current work part
        Dim fileName As String = IO.Path.GetFileName(workPart.FullPath)
        Dim fileNameNoExt As String = IO.Path.GetFileNameWithoutExtension(workPart.FullPath)
        Dim parentFolder As String = IO.Path.GetDirectoryName(workPart.FullPath)
        Dim root As String = IO.Path.GetPathRoot(workPart.FullPath)

        'use listing window to write to file and window
        lw.SelectDevice(ListingWindow.DeviceType.FileAndWindow, outputFile)
        lw.Open()

        For Each temp As Expression In workPart.Expressions
            'lw.WriteLine("name: " & temp.Name)

            Select Case temp.Type
                Case Is = "Number"
                    Try
                           lw.WriteLine(temp.equation)
                                Catch ex As NullReferenceException
                        lw.WriteLine("expression is constant (unitless)")
                    Catch ex2 As Exception
                        lw.WriteLine("!! error: " & ex2.Message)
                    End Try
                Case Is = "String"
                    lw.WriteLine("string value: " & temp.StringValue)
                Case Is = "Integer"
                                lw.WriteLine(temp.equation)
                Case Is = "Point"
                    lw.WriteLine("point value: " & temp.PointValue.ToString)
                Case Else
                    lw.WriteLine("Type: " & temp.Type & " is not handled by this journal")
             End Select
            'lw.WriteLine("")
        Next

        lw.Close()
        'flush file buffer by changing listing window device
        lw.SelectDevice(ListingWindow.DeviceType.Window, ""

    End Sub
```

Figure 53: NX Journal file for exporting expressions

5.7.5 Training for object detection

One main function of the Space Factory 4.0 is that a new module has to be assembled in the Space Factory 4.0. Therefore, it would be useful if the robotic arm can detect which kind of object is in front of him. A Convolutional Neural Network (as described in 4.3.5) is trained to detect different classes of objects. The CNN chosen is a predefined model from TensorFlow [118]. It is a Faster R-CNN architecture with Inception V2 feature extractor (for comparison see chapter 4.3.5). It can be operated in real-time but with slow frames per second. But it has a higher accuracy if compared to the MobileNet architecture. To train the model a set of around 30 pictures per class has been taken and labeled with bounding boxes. The open-source tool used is called LabelImg by tzutalin [119]. There a bounding box is drawn by hand around the desired object and labeled with a unique name. The process is shown in Figure 54 with the drawn bounding box in green. Afterwards a name was defined for this class called "cubesat_module". Two other classes were labeled additionally a picture of an onboard computer, screenshots from a RaspberryPi CAD model and a picture of a UFO. The number of classes can be expanded infinite but on the other side the amount of human resources for the labeling process increases dramatically.

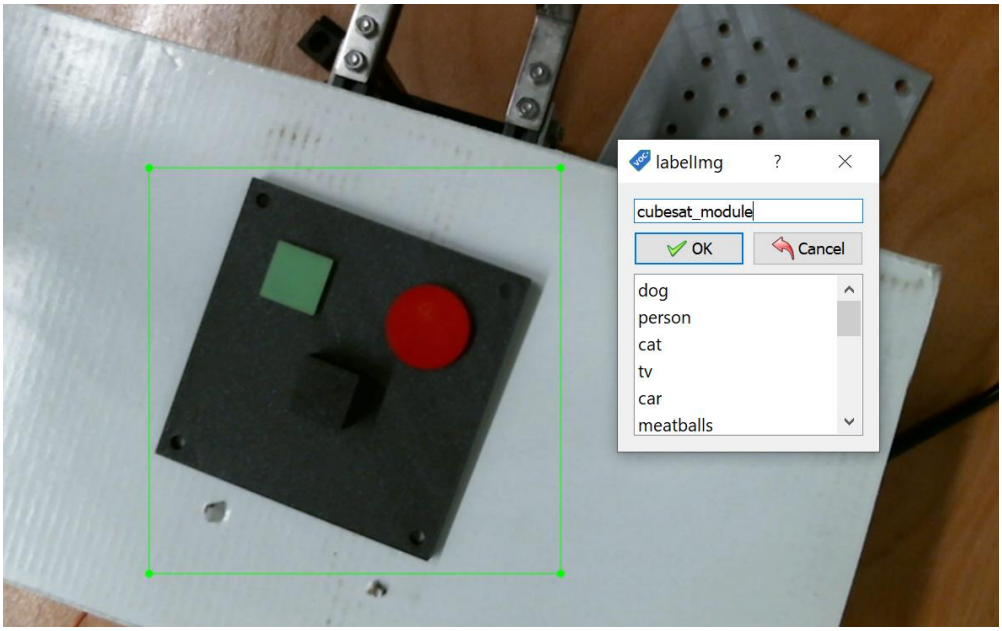


Figure 54: Labeling process of the CubeSat Module

Then the labeled data is exported into an XML file. This whole process is very time-consuming, so it was limited to 30 pictures per class. In a future Space Factory 4.0, this process could be automated and integrated into the assembly process. A higher number of pictures per class would be realistic for the implementation of the Space Factory 4.0 but not possible in this thesis. After the labeling process the CNN can be trained with the images and the XML file.

5.7.6 Region based Convolutional Neural Network

For this object detection system, a further developed Convolutional Neural Network is used. The so-called Region based Convolutional Network (R-CNN). Developed by R. Girshick in 2014 [120], the network uses selective search to extract only 2000 regions from the image and is defined as region proposals. Afterwards a CNN performs forward computation to extract features from each region proposal. At last, the features of each region proposal are used to predict their category and bounding boxes. One year later the same author developed the so

called Fast R-CNN. The main performance issue of the old R-CNN model is the need to independently extract features for each proposed region. It cannot perform real-time tasks because the computational time takes around 47 seconds. Using Fast R-CCN, instead of feeding the region proposals to the CNN, the input image is fed to the CNN and generates a convolutional feature map. From the convolutional feature map regions of proposals are identified and warped into square. A pooling layer reshapes them into a fixed size so that it can be fed into a fully connected layer. Fast R-CNN is faster than R-CNN because not 2000 region proposals are fed to the convolutional neural network every time. Instead, the convolution operation is performed only once per image and a feature map is generated from it. Fast R-CNN can achieve almost real-time rates when ignoring the spending time on region proposals.[121] The architecture used in this thesis is called Faster R-CNN developed by S. Ren [122]. Faster R-CNN replaces selective search with a region proposal network. Selective search is a slow and time-consuming process affecting the performance of the network. The region proposal network is used to predict the region proposals. The predicted region proposals are then reshaped using a RoI pooling layer, which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes. This reduces the number of proposed regions generated while ensuring precise object detection. The architecture of the Faster R-CNN is shown in Figure 55.

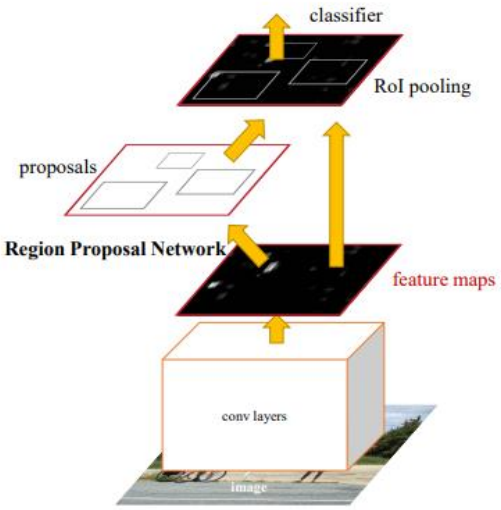


Figure 55: Region-based convolutional neural network [122]

The Faster R-CNN is much faster than its predecessors. Enabling it to be used for real-time object detection with a frame rate of 5 fps. While most region proposal methods used in research are implemented on the CPU, the Faster R-CNN takes advantage of the GPU. In the ImageNet competitions Faster R-CNNs are the basis for several 1st place winners. [122]

For the implementation of a Faster R-CNN, the library object detection API from TensorFlow is used. [118]. Tensorflow provides several pre-trained classifiers with specific neural network architectures. As described in chapter 4.3.6, the different architectures vary in processing speed, accuracy and training time. The pre-trained models included in TensorFlow are mainly the architectures: SSD, MobileNet, Faster R-CNN and R-FCN. For this thesis, the pre-trained model “faster_rcnn_inception_v2_coco” is selected, which uses the Faster R-CNN with the Inception V2 feature extractor. Stated by TensorFlow, it has a processing speed of 58 ms and an accuracy of 28 mAP on the COCO dataset. [118] The training time on TensorFlow is around 4-6 h depending if it operates on CPU or GPU. The training was conducted until the loss function sufficiently dropped under 0.05 as shown in Figure 56. On the y-axis the total loss is plotted,

which is the sum of the classification loss and the localization loss. The x-axis represents the time steps. On a CPU the processing speed was only around 5 sec/step.

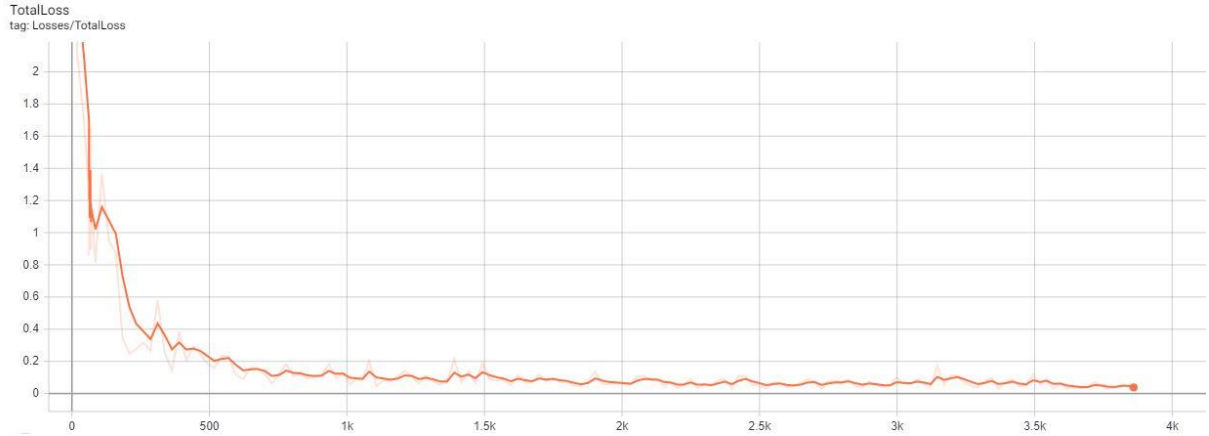


Figure 56: Total Loss Rate (exported from TensorBoard)

In general, the calculation of the loss value for a linear classifier can be described as [123]:

$$L = \frac{1}{N}\sum_{i} L_i(f(x_i, W), y_i)$$

With a given dataset of examples $\{x_i, y_i\}_{i=i}^{N}$, where x_i is the image and y_i the label as an integer of its corresponding class. W is the score of the prediction of the class.

```
# Name of the directory containing the object detection module
MODEL_NAME = 'faster_rcnn_inception_v2_coco_2018_01_28'
# path to current working directory
CWD_PATH = 'C:/tensorflow1/models/research/object_detection'
# Path to frozen detection graph .pb file, which contains the model that is used
# for object detection.
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph2.pb')
# Path to label map file
PATH_TO_LABELS = os.path.join(CWD_PATH,'training','labelmap2.pbtxt')
# Number of classes the object detector can identify
NUM_CLASSES = 4
## Load the label map.
# Label maps map indices to category names, so that when the convolution
# network predicts `1`, it knows that this corresponds to `cubesat_module`.
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map, max_num_classes=NUM_CLASS
category_index = label_map_util.create_category_index(categories)
# Define input and output tensors (i.e. data) for the object detection classifier
# Input tensor is the image
image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
# Output tensors are the detection boxes, scores, and classes
# Each box represents a part of the image where a particular object was detected
detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
# Each score represents level of confidence for each of the objects.
# The score is shown on the result image, together with the class label.
detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')
detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')
# Number of objects detected
num_detections = detection_graph.get_tensor_by_name('num_detections:0')
# Initialize webcam feed
video = cv2.VideoCapture(0, cv2.CAP_DSHOW)
video.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)
video.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
```

Figure 57: Extract from the object detection code

After the training process a frozen inference graph is created. This frozen model is imported for the object detection process. An extract of the code is shown in Figure 57. It is mainly based on code provided by the TensorFlow Object Detection API [118]. After the importing process the webcam is initialized and the detection process can be started. Boxes, scores and labels are drawn around the detected object in each frame of the video.

5.7.7 Quality 4.0

Another subcategory of Industry 4.0 is Quality 4.0. To achieve Quality 4.0, one of the key things is the availability of real-time data. This real-time data will be provided by the Computer Vision system. But in general, all kinds of sensors can be implemented for quality inspection. Furthermore, digital twins, as a digital representation of the physical object or other simulations can be included in the quality analysis. By allowing to aggregate data continuously, these technologies enable preventive maintenance and optimized production. The likelihood of releasing poor quality products will be reduced by Quality 4.0.

5.7.8 Quality Inspection

After the detection with the Neural Network, a picture is taken of the CubeSat module and a simplified quality inspection is executed. Several mathematical filters are applied to the picture for further analyzing and measurements. The analyzing process can be summarized into these following steps:

1. Check if colored parts (red/green) are present
2. Convert to grayscale image
3. Blur the image with Gaussian filer
4. Canny edge detection
5. Erode and dilate function (to close gaps between object edges)
6. Draw Bounding Box between midpoints of edges
7. Compute the size of the object with pixel to metric conversion

First of all, the CubeSat module will be checked if the parts red circle and green rectangle are present. The color number of the digital twin will be read from the expression file, which is saved in the Centralized Information Base. For analyzing the picture, the open-source library OpenCV (Open Source Computer Vision Library) will be used. [112] The library has more than 2500 optimized algorithms for Computer Vision, which be displayed in the following text with “cv2”. The full code is available in the appendix. The presence of the part will be determined by applying a color filter of the color of the part. If not enough color points, e.g. red points, are counted, a warning will be printed that the part is missing. If the color is different on the digital twin, a warning will also be printed. If both color checks are successful, a boolean value will be returned that the check was successful. Afterwards the measure sequence is started. First, the picture is converted to grayscale (cv2.cvtColor), to smoothen the picture a Gaussian filter is applied (cv2.GaussianBlur). Afterwards the function cv2.Canny is executed. This function finds edges in an image using the algorithm invented by John F. Canny in 1986 [124]. At the end, the morphological operations dilation and erosion are used to close the gaps between the detected edges and remove noise (cv2.dilate, cv2.erode). The final result is shown in Figure 58 (left).

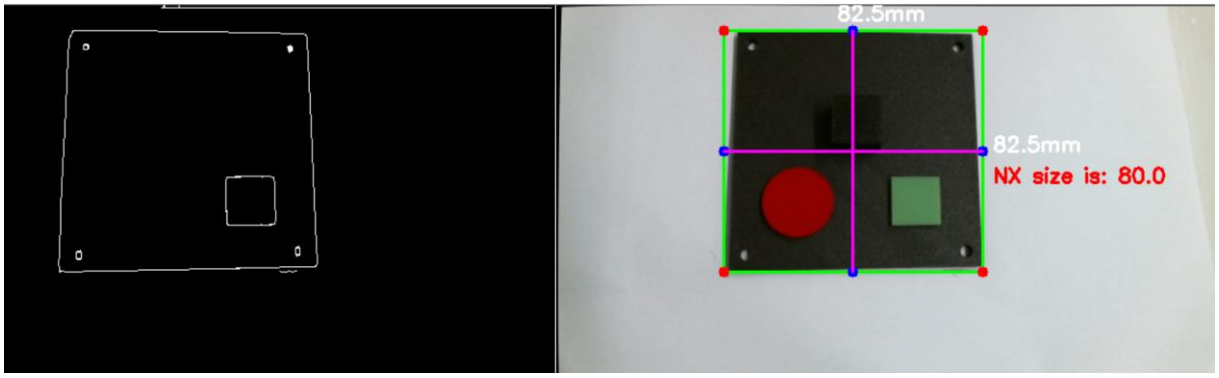


Figure 58: Canny edge detection (left) and size measurement (right)

This map of edges can be used to measure the object. With the function cv2.findContours, which saves the x and y coordinates of the contour. A contour is a curve joining all the continuous points, having the same color intensity. With the x and y points the pixel distance of the contour can be calculated. With this information and right pixel to metric ratio, the dimension in millimeter can be approximated. The camera is always in a fixed position and the distance to the measured object is also fixed. The unique pixel to metric ratio can be determined with a known object and its size.

$$Pixel\ Metric\ Ratio = \frac{Distance\ (Pixel)}{Distance\ (mm)}$$

The determined dimension will be displayed and printed on the taken image, as shown in Figure 58 (right). The dimension of the digital twin is imported from the CIB and also shown with “NX size is: “. Afterwards the red and green parts are also measured.

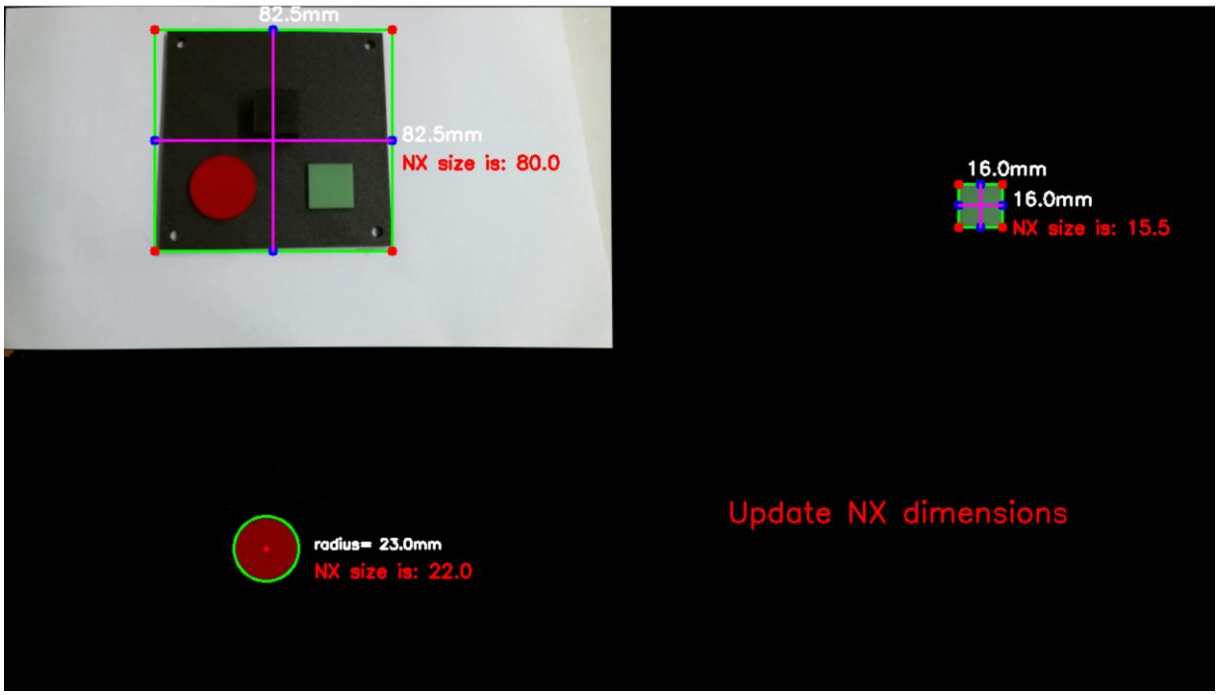


Figure 59: Visualization of the measured parts

For measuring the red and green parts the filtered picture of the first step is used. Where it was checked if the parts are present. For the green rectangle the same measurement procedure as the CubeSat module can be conducted. But for the red circle the Canny edge detection is not effective enough because the bigger circle is filtered out. Therefore, the Hough Circle Transform function (cv2.HoughCircles) is performed. It can detect circles in a picture and return their radius. Afterwards the dimensions of the digital twin are imported from Siemens NX. Then they are compared with the physical object, as shown in Figure 59.

The measured dimension is updated to the digital twin, which is visualized with ("Update NX dimensions"). The expression file from the CIB is overwritten with the measured values and will be saved as an instance of the digital twin. This file is imported into Siemens NX with NX journal to update the expression. The import code is similar to the export code described in chapter 5.7.4 and can be found in the appendix. Now the digital twin represents the object with the dimensions of the physical world. The updated digital twin is shown in Figure 60.

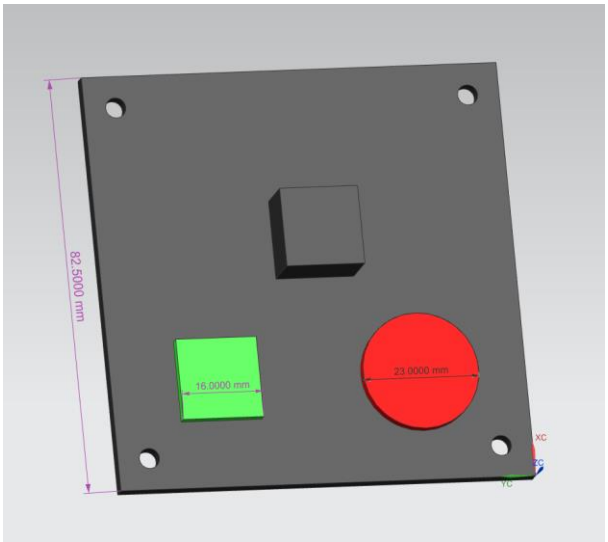


Figure 60: Updated dimension on the digital twin

To make this instance of the digital twin unique in the Space Factory 4.0 system and XML file is created of this digital twin. The ID includes the name of the object and date and time of the creation.

ID:	**DD**	**MM**	**YYYY**	**HH**	**MM**	**SS**
	Day	Month	Year	Hours	Minutes	Seconds

An example of the ID is shown in the XML file in Figure 61. Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is readable for human and also readable for machines. The file also includes the relevant expressions of the digital twin.

```
<?xml version='1.0' encoding='utf-8'?>
<elements>
    <cubesatmodule>
        <ID>cubesatmodule08052020111425</ID>
        <expressions>
                <cubesat_length>82.5</cubesat_length>
                <cubesat_color_number>173</cubesat_color_number>
                <circle_color_number>186</circle_color_number>
                <diameter>23</diameter>
                <rectangle_length>16</rectangle_length>
                <rectangle_color_number>29</rectangle_color_number>
        </expressions>
    </cubesatmodule>
</elements>
```

Figure 61: Example of the unique ID in the XML file

5.7.9 Robot control

After the quality inspection and comparison with the digital twin the CubeSat module will be integrated into the CubeSat. Therefore, a simple pick and place operation is developed. The sequence of the robotic arm is shown in Figure 62. Step 1 is the start position of the robot. In step 2 the robotic arm moves down and closes the gripper. In step 3 the robotic arm picks up the CubeSat module. Afterwards it rotates to the desired position, moves the robotic arm down and opens the gripper. The sequence ends after Step 4.

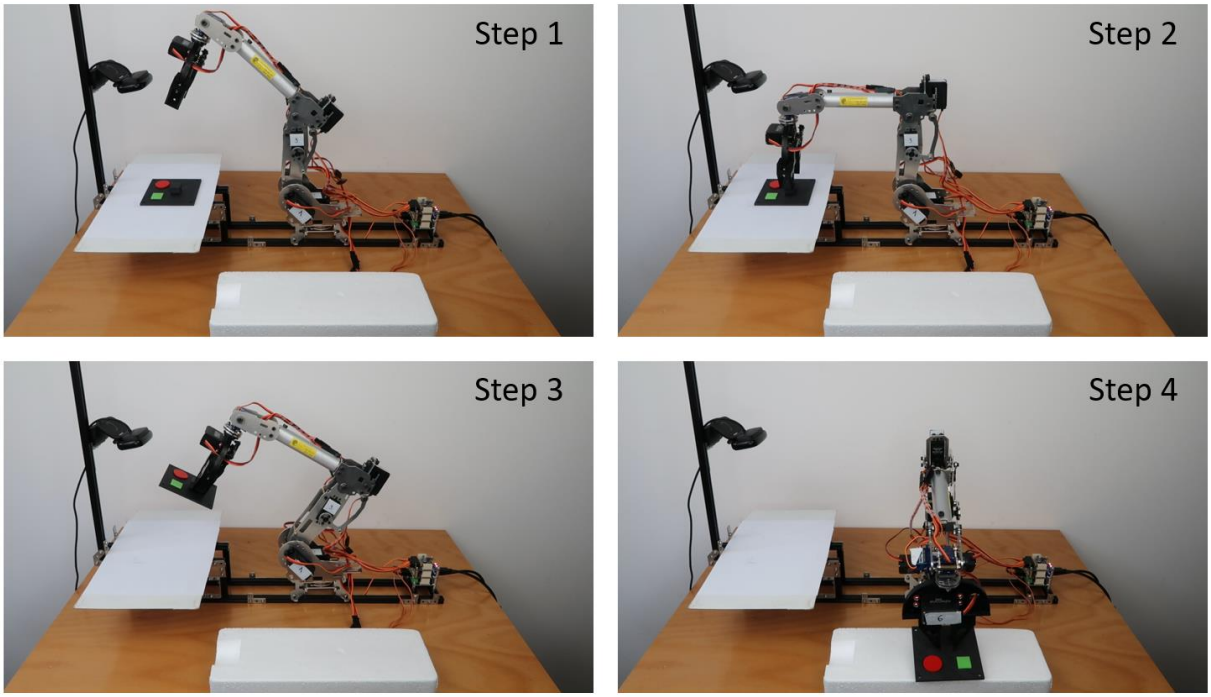


Figure 62: Sequence of the robot

6 Evaluation

The Region-based Convolutional Neural Network, described in chapter 5.7.6, could successfully be implemented in the Space Factory 4.0. The trained objects are detectable with high accuracy. The detection process is performing in a real-time video. However, the frame rate is quite low. Multiple objects are also detectable and separable, as shown in Figure 63 (left).

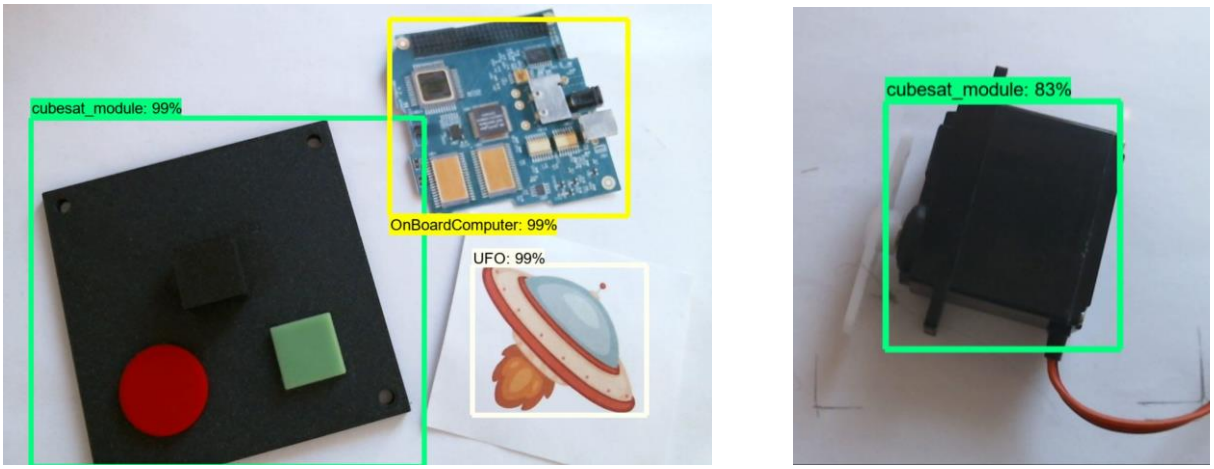


Figure 63: Multiple Objects detectable (left) and false-positive case (right)

The detection process can still be optimized. As shown on the right in Figure 63, not trained objects (e.g. servo) are assigned to a class. This error is called a false positive detection. In this case, the servo is detected as the class CubeSat module. This can be attributed to several reasons. First of all, the CubeSat module and the servo look similar. They are mainly represented by large black areas and have a rectangle shape. The R-CNN is only trained on the pictures of the CubeSat module, an UFO, an On-Board-Computer and the Raspberry Pi. Hence, its decision making is very limited only to these four trained classes. It only tries to find these four objects in the field of view of the system. For example, the class servo is not trained on this object detector, so it is not able to classify the object in front correctly. The class CubeSat module will be predicted instead. In general, the data for this object detector is very limited. Each class was only trained on approximately 30 pictures. This number is very low for an object detector. Common object detectors are trained by the COCO Dataset [101]. It consists of over 200.000 images with around 80 classes. This is an average of 2500 images per class. So the size of the training data used in this thesis is not comparable. The main reason the training data is so low is because the desired objects are not available in the COCO dataset. The picture taking and labeling process by hand is very time consuming, as described in chapter 5.7.5. If this process can be automated, it would be a great improvement. Therefore, the idea of training the R-CNN with a rendered CAD model was also researched. The idea is to directly use a digital twin, as a real-world representation, for training. As shown in Figure 64, on the left is the training data for the object detector and on the right is the real-world object. The digital twin is different in color and some parts are missing. For example, the color of the printed circuit board is only monochrome in green. On the real object, it is colored in different shades of green. Even with such major differences in the digital model and the real object, the object detector was able to detect its class correctly. As shown on the right in Figure 64, it is able to predict the correct class with 87% confidence. On the other side, when changing the position of the real object or lightning condition, it predicted the class On-Board-Computer. This misconception occurs probably because the Raspberry Pi and the On-Board-Computer consist of similar elements, such as transistors, blocks, pins and other electronic components.

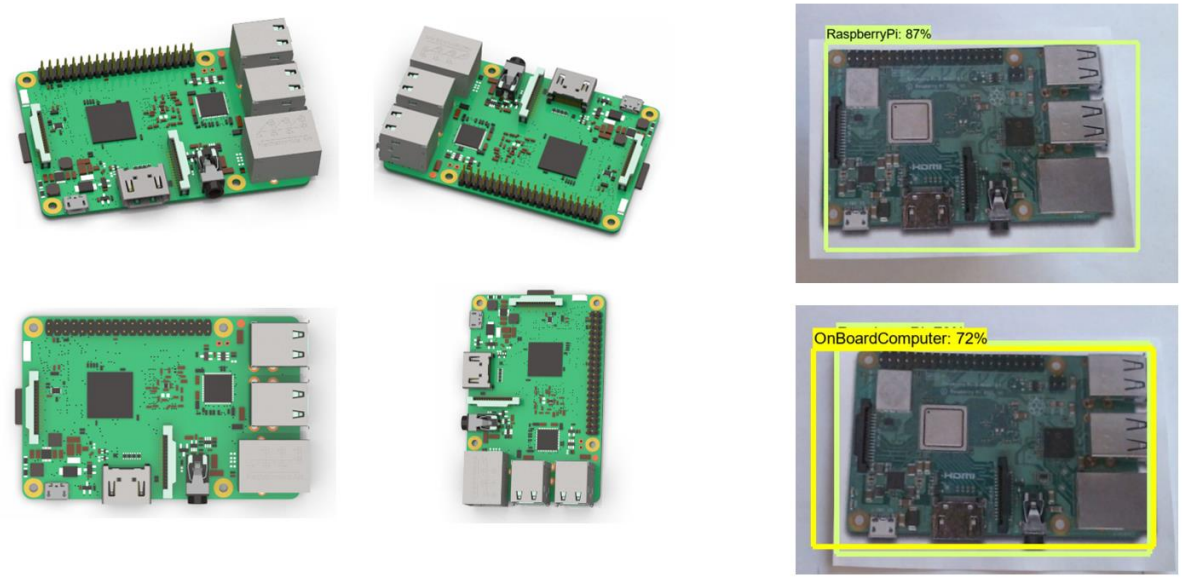


Figure 64: Training data of the Rasberry Pi CAD model (left) and real-world object (right)

If the digital twin would be of higher accuracy and parts designed in the right color, it would probably detect the object with a higher prediction. Moreover, the false-positive rate would be much lower.
Not only complex algorithms such as Convolutional Neural Network have been investigated. For simple color detection, a neural network is not necessary. In this case, the traditional machine vision algorithms show promising results. For monochrome parts, a color filter is applied to detect the presence or absence of a part. An example is shown in Figure 65. In this example, the red part is not assembled on the CubeSat module. The information of the part is imported from the CIB and a warning is displayed that the check was unsuccessful. Now the CubeSat module can be fixed or removed from the assembly process.

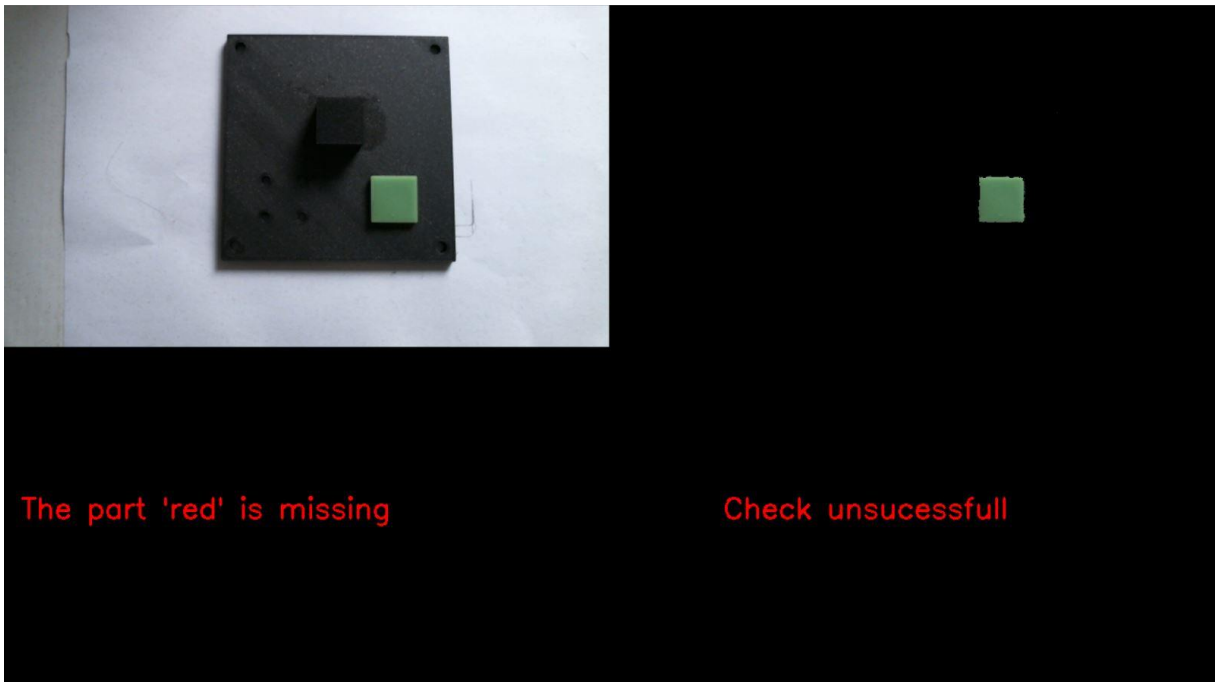


Figure 65: Detection of absence or presence of monochrome part

7 Future Prospect

The evaluation shows that the Computer Vision system for the Space Factory 4.0 has promising features, which will be discussed in the following chapter. First, the object detection system with the Convolutional Neural Network has been established and can be further improved in the future. Only one relevant part, the simplified CubeSat module, has been trained to the neural network. Other relevant objects could be trained to this object detection system to enhance and improve the prediction quality. To achieve this, more pictures need to be taken and added to the database. Second, the possible use of rendered CAD data was only investigated in a small scale. A detailed and fully rendered CAD model could be created of a CubeSat module. Pictures of this rendered CAD model can then be used to train the neural network [125]. Third, only one type of neural network was used for the object detection system. The comparison shows that other neural network architectures show promising usage. To have full control of the object detection process, an own neural network could also be developed. In this case, it could also be investigated if features of the digital twin (CAD model) can be directly used to train the neural network. For example, the information of an edge is already saved in the CAD model and can maybe be passed on to the neural network.

The final goal of the object detection system would be to verify the whole assembly process. That can contain all electronic parts (switches, resistors, etc.), which are mounted on the CubeSat module, as shown in Figure 66. On a bigger scale, it can also be inspected if one CubeSat module is correctly installed in the CubeSat.

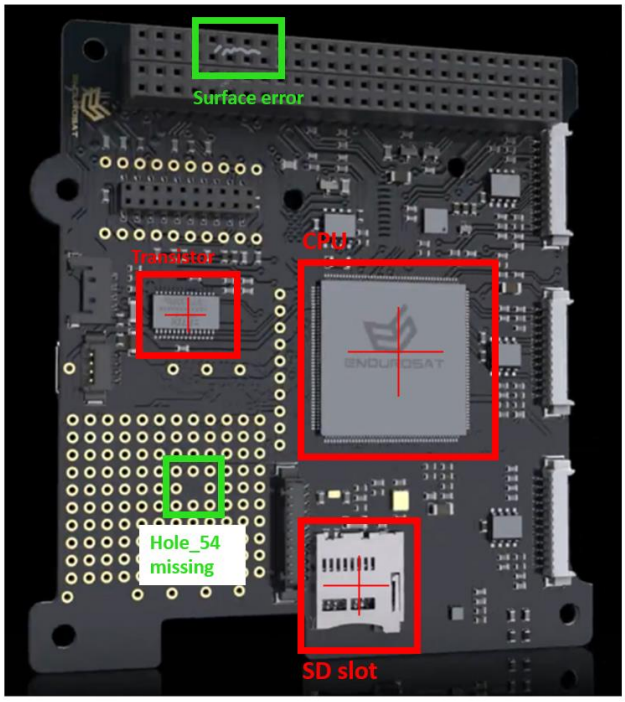


Figure 66: Assembly verification of CubeSat module

The developed Computer Vision system is an entry point for many other AI applications. One example would be for the Self-Learning Production System, also introduced in this thesis. For receiving feedback from the environment, this developed object detection system is helpful. Because of the scope of the self-learning project, it was only introduced theoretically. Built upon this knowledge and with the object detection system already in place, a Self-Learning Production System can be implemented in the future. It would also be necessary to develop a

digital twin of the robotic arm on a different platform, because the Siemens NX platform is not integrable enough.

This thesis also researched the potential use of predictive maintenance. It successfully solved one simple classification problem with supervised learning. Because of the scope of this thesis, only one algorithm, the Support Vector Machine, was implemented. In the field of supervised and also unsupervised learning, there are many more algorithms available which can be compared. More classification, association or clustering problems of the Space Factory 4.0 have to be identified so that these machine learning algorithms can be applied. In generating more useful data, the simulation of the digital twin can also be used.

8 Conclusion

This thesis aimed to integrate artificial intelligence with the digital twin in the Space Factory 4.0. Therefore, different relevant technologies and achievements in the field of artificial intelligence and Industry 4.0 are showcased. The technologies focused on are: Predictive Maintenance, Self-learning Production System and Computer Vision.
For a better understanding of these systems, the fundamentals of artificial intelligence and the history of artificial intelligence are introduced in the beginning. Subsequently, the different sub-branches of AI are presented. In this thesis, the focus lies on Computer Vision, robotic and machine learning. The branches of machine learning, supervised, unsupervised and reinforcement learning are further described.
For Predictive Maintenance, the implementation process is described and several different algorithms in supervised and unsupervised learning are elaborated. Afterwards the Support Vector Machine algorithm is implemented in the Space Factory 4.0. This algorithm can be used for simple classification problems. The implementation includes the analysis of the servo motor of the robotic arm. The data of the servo motor was measured and the algorithm trained. In the end, the algorithm can classify two situations of the servo motor. If high force is applied to the servo motor, it is classified as a “NOK” case and if no force is applied, it is classified as an “OK” case.
Also for the Self-Learning Production System different research and relevant projects are showcased. In this regard, reinforcement learning is described in detail, which is the theory behind a self-learning system. Afterwards different requirements are derived from these projects and also discussed how these features could be implemented in the Space Factory 4.0. It is also shown which role the digital twin has in this field of a self-learning system.
At last, the integration of Computer Vision in the framework of Space Factory 4.0 is researched. First, the traditional machine vision is compared to the new deep learning method. After explaining both technologies in detail, a concept for Computer Vision is presented. Therefore, a system is developed to synchronize the digital twin with the real physical object. A Convolutional Neural Network is trained and added to detect the physical objects. Afterwards machine vision algorithms are applied to extract information from the physical object and synchronized with the digital twin. Color and dimensions can now be synchronized between both entities.

All mentioned artificial intelligence technologies show promising benefits for the Space Factory 4.0. The research in artificial intelligence is changing fast in the present time and the systems become more and more complex. Artificial intelligence is not one single, unified technology. AI is a set of interrelated technology components that can be used in a wide variety of combinations depending on the problem it addresses. The fuel of AI is the training data it receives. This thesis showed that digital twins can be a useful tool to generate these data for future AI applications and that the Centralized Information Base is an organized platform to save these data.

9 References

[1] A. Mann, *Starlink: SpaceX's satellite internet project.* [Online]. Available: https://www.space.com/spacex-starlink-satellites.html (accessed: Apr. 28 2020).

[2] J. Bort, *Jeff Bezos explains why he's trying to colonise the moon: 'We need to go to the moon to save the Earth'.* [Online]. Available: https://www.businessinsider.com.au/amazon-ceo-jeff-bezos-colonize-moon-2019-6?r=US&IR=T (accessed: Apr. 28 2020).

[3] R. Anderl and T. Weber Martins, "Space Factory 4.0 – Wie die Ansätze der Industrie 4.0 zukünftigen In-Orbit-Plattformen für die robotische Montage von hochmodularen Satelliten vorantreiben können?," DGLR Kongress, 2018.

[4] T. Weber Martins and R. Anderl, "DIGITAL TWINS FOR SPACE FACTORY 4.0," Department of Computer Integrated Design, 2019.

[5] Bundesministerium für Wirtschaft und Energie (BMWi) Indutrie 4.0, "Technologieszenario „Künstliche Intelligenz in der Industrie 4.0"," 2019.

[6] J. McCarthy, "WHAT IS ARTIFICIAL INTELLIGENCE?," Stanford, 2007. Accessed: Mar. 11 2020. [Online]. Available: http://jmc.stanford.edu/articles/whatisai.html

[7] The National Science and Technology Council, "PREPARING FOR THE FUTURE OF ARTIFICIAL INTELLIGENCE," 2016.

[8] M. Negrotti, *Understanding the Artificial: On the Future Shape of Artificial Intelligence*. London: Springer, 1991.

[9] A. Nazre and R. Garg, *A Deep Dive in the Venture Landscape of AI.* [Online]. Available: https://medium.com/@aniruddhanazrelive/a-deep-dive-in-the-venture-landscape-of-ai-ajit-nazre-rahul-garg-de0543231eb2 (accessed: Mar. 11 2020).

[10] K. Frankish and W. M. Ramsey, Eds., *The Cambridge handbook of artificial intelligence*. Cambridge: Cambridge University Press, 2014.

[11] J. L. McClelland and D. E. Rumelhart, *Parallel distributed processing: Explorations in the microstructure of cognition,* 9th ed. Cambridge, Mass.: MIT Pr, 1989.

[12] X. Yao, J. Zhou, J. Zhang, and C. R. Boer, "From Intelligent Manufacturing to Smart Manufacturing for Industry 4.0 Driven by Next Generation Artificial Intelligence and Further On," in *Industrial digitalization by enterprise systems: Proceedings : 2017 5th International Conference on Enterprise Systems : ES 2017 : 22-24 September 2017, Beijing, China*, Beijing, 2017, pp. 311–318.

[13] W. Ertel, *Grundkurs Künstliche Intelligenz: Eine praxisorientierte Einführung,* 4th ed. Wiesbaden: Springer Vieweg, 2016.

[14] T. Ormston, *Time delay between Earth and Mars.* [Online]. Available: http://blogs.esa.int/mex/2012/08/05/time-delay-between-mars-and-earth/ (accessed: Feb. 3 2020).

[15] D. Gaines, *M2020.* [Online]. Available: https://www-aig.jpl.nasa.gov/public/projects/m2020-scheduler/

[16] S. Chien, *ASE.* [Online]. Available: https://ase.jpl.nasa.gov/

[17] JPL, *JPL Artifical Intelligence Group Projects.* [Online]. Available: https://www-aig.jpl.nasa.gov/public/projects/

[18] ACT, *ACT artficial intelligence projects.* [Online]. Available: https://www.esa.int/gsp/ACT/research/ai_main.html

[19] ESA, *Robots in space.* [Online]. Available: https://www.esa.int/Enabling_Support/Preparing_for_the_Future/Discovery_and_Preparation/Robots_in_space2 (accessed: Feb. 3 2020).

[20] DLR, *CIMON - Der intelligente Astronautenassistent.* [Online]. Available: https://www.dlr.de/content/de/artikel/news/2018/1/20180302_cimon-der-intelligente-astronautenassistent_26307.html (accessed: Feb. 3 2020).

[21] H. Kesuma, S. Ahmadi-Pour, A. Joseph, and P. Weis, "Artificial Intelligence Implementation on Voice Command and Sensor Anomaly Detection for Enhancing

Human Habitation in Space Mission," in *2019 9th International Conference on Recent Advances in Space Technologies (RAST)*, Istanbul, Turkey, 2019?, pp. 579–584.
[22] Airbus, *Astronaut assistant CIMON-2 is on its way to the International Space Station.* [Online]. Available: https://www.airbus.com/newsroom/press-releases/en/2019/12/astronaut-assistant-cimon2-is-on-its-way-to-the-international-space-station.html (accessed: Feb. 3 2020).
[23] D. Izzo, M. Märtens, and B. Pan, "A survey on artificial intelligence trends in spacecraft guidance dynamics and control," *Astrodyn*, vol. 3, no. 4, pp. 287–299, 2019, doi: 10.1007/s42064-018-0053-6.
[24] C. Sánchez-Sánchez and D. Izzo, "Real-time optimal control via Deep Neural Networks: study on landing problems," Oct. 2016. [Online]. Available: http://arxiv.org/pdf/1610.08668v1
[25] Chu, X., Alfriend, K. T., Zhang, J., Zhang, Y, "Q-learning algorithm for path-planning to maneuver through a satellite cluster," Proceedings of 2018 AAS/AIAA Astrodynamics Specialist Conference, 2018.
[26] Thiago Weber Martins Aaron Pereira, Thomas Hulin, Oliver Ruf, Stefan Kugler, Alessandro M. Giordano, Ribin Balachandran, Fabian Benedikt, John Lewis, Reiner Anderl, Klaus Schilling, Alin Albu-Schäffer, "Space Factory 4.0 - New processes for the robotic assembly of modular satellites on an in-orbit platform based on „Industrie 4.0" approach," 69th International Astronautical Congress (IAC), Bremen, 2018.
[27] A. Roth, Ed., *Einführung und Umsetzung von Industrie 4.0: Grundlagen, Vorgehensmodell und Use Cases aus der Praxis*. Berlin, Heidelberg: Springer Gabler, 2016.
[28] R. Anderl, "Vernetzte Produktentestehungsprozesse," 2018.
[29] A. Ustundag and E. Cevikcan, *Industry 4.0: Managing the digital transformation*. Cham: Springer, 2018.
[30] B. Bagheri, S. Yang, H.-A. Kao, and J. Lee, "Cyber-physical Systems Architecture for Self-Aware Machines in Industry 4.0 Environment," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 1622–1627, 2015, doi: 10.1016/j.ifacol.2015.06.318.
[31] M. Grieves, "Digital Twin: Manufacturing Excellence through Virtual Factory Replication," 2014.
[32] M. Grieves, "Origins of the Digital Twin Concept," 2016.
[33] E. Glaessgen and D. Stargel, "The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles," in *Structures, Structural Dynamics, and Materials and Co-located Conferences: 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, Honolulu, Hawaii, 2012.
[34] T. Kuhn, "Digitaler Zwilling," *Informatik Spektrum*, vol. 40, no. 5, pp. 440–444, 2017, doi: 10.1007/s00287-017-1061-2.
[35] R. Anderl, S. Haag, K. Schützer, and E. Zancul, "Digital twin technology – An approach for Industrie 4.0 vertical and horizontal lifecycle integration," *it - Information Technology*, vol. 60, no. 3, pp. 125–132, 2018, doi: 10.1515/itit-2017-0038.
[36] T. Bauernhansl, "Wgp-Standpunkt Industrie 4.0," 2016.
[37] U. Bracht, D. Geckler, and S. Wenzel, *Digitale Fabrik: Methoden und Praxisbeispiele,* 2nd ed. Berlin: Springer Vieweg, 2018.
[38] IERC, *Internet of Things.* [Online]. Available: http://www.internet-of-things-research.eu/about_iot.htm (accessed: Mar. 9 2020).
[39] Plattform Industrie 4.0, "Alignment Report for Reference Architectural Model for Industrie 4.0/ Intelligent Manufacturing System Architecture," 2018. Accessed: Mar. 9 2020. [Online]. Available: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/hm-2018-manufactoring.html
[40] Plattform Industrie 4.0, "RAMI 4.0 – Ein Orientierungsrahmen für die Digitalisierung," 2018. Accessed: Mar. 9 2020. [Online]. Available: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/rami40-einfuehrung-2018.html

[41] T. Gamer, M. Hoernicke, B. Kloepper, R. Bauer, and A. J. Isaksson, “The Autonomous Industrial Plant -Future of Process Engineering, Operations and Maintenance,” *IFAC-PapersOnLine*, vol. 52, no. 1, pp. 454–460, 2019, doi: 10.1016/j.ifacol.2019.06.104.
[42] B. Langefeld, *AUTONOMOUS PRODUCTION - NEW OPPORTUNITIES THROUGH AI?* [Online]. Available: https://www.rolandberger.com/sv/Point-of-View/Autonomous-production-New-opportunities-through-Artificial-Intelligence.html (accessed: Mar. 10 2020).
[43] V. L. Pisacane, Ed., *Fundamentals of space systems,* 2nd ed. Oxford: Oxford Univ. Press, 2005. [Online]. Available: http://www.loc.gov/catdir/enhancements/fy0635/2003024375-d.html
[44] *ECSS-E-ST-10-03C*, 2012.
[45] The CubeSat Program, Cal Poly SLO, “CubeSat Design Specification Rev. 13,” 2014.
[46] A.-H. Jallad, P. Marpu, Z. Abdul Aziz, A. Al Marar, and M. Awad, “MeznSat—A 3U CubeSat for Monitoring Greenhouse Gases Using Short Wave Infra-Red Spectrometry: Mission Concept and Analysis,” *Aerospace*, vol. 6, no. 11, p. 118, 2019, doi: 10.3390/aerospace6110118.
[47] NASA, “CubeSat101 Basic Concepts and Processes for First-Time CubeSat Developers,” 2017.
[48] Unisec Europe, “CubeSat Subsystem Interface Definition,” 2017.
[49] W. Ley, K. Wittmann, and W. Hallmann, Eds., *Handbuch der Raumfahrttechnik,* 5th ed. München: Hanser, 2019.
[50] J. S. Gero and F. Sudweeks, *Artificial Intelligence in Design ’96*. Dordrecht: Springer Netherlands, 1996.
[51] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. Cambridge, Mass., London: The MIT Press, 2012.
[52] C. M. Bishop, *Pattern recognition and machine learning,* 8th ed. New York, NY: Springer, 2009.
[53] Cognex Corporation, “Deep Learning vs. Machine Vision,” One Vision Drive Natick, MA 01760 USA, 2019.
[54] S. Wang, W. Chaovalitwongse, and R. Babuska, “Machine Learning Algorithms in Bipedal Robot Control,” *IEEE Trans. Syst., Man, Cybern. C*, vol. 42, no. 5, pp. 728–743, 2012, doi: 10.1109/TSMCC.2012.2186565.
[55] H. LI, *Which machine learning algorithm should I use?* [Online]. Available: https://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/ (accessed: Mar. 13 2020).
[56] Gregory Piatetsky-Shapiro, “Discovery, Analysis, and Presentation of Strong Rules,” in *Knowledge Discovery in Databases*, 1991.
[57] C. C. Aggarwal, *Neural Networks and Deep Learning*. Cham: Springer International Publishing, 2018.
[58] Argility, *All the solutions you need to engage your customers wherever they are.* [Online]. Available: https://www.argility.com/argility-ecosystem-solutions/iot/machine-learning-deep-learning/ (accessed: Mar. 27 2020).
[59] R. Gouriveau, K. Medjaher, and N. Zerhouni, *From prognostics and health systems management to predictive maintenance 1: Monitoring and prognostics*. Hoboken, NJ: Wiley, 2016. [Online]. Available: http://onlinelibrary.wiley.com/book/10.1002/9781119371052
[60] *DIN EN 13306 Maintenance - Maintenance terminology*, 2017.
[61] I. MathWorks, *Designing Algorithms for Condition Monitoring and Predictive Maintenance.* [Online]. Available: https://in.mathworks.com/help/predmaint/gs/designing-algorithms-for-condition-monitoring-and-predictive-maintenance.html (accessed: Jan. 30 2020).
[62] P. Jahnke, “Machine Learning Approaches for Failure Type Detection and Predictive Maintenance,” TU Darmstadt, 2015.

[63] L. Rokach and O. Z. Maimon, *Data mining with decision trees: Theory and applications*. Singapore: World Scientific, 2008. [Online]. Available: http://site.ebrary.com/lib/academiccompletetitles/home.action
[64] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge: Cambridge University Press, 2014.
[65] T. Zhang, "Predicitve Maintenance with Sensors in Utilities," 2015. Accessed: Apr. 8 2020. [Online]. Available: https://www.slideshare.net/TinaZhang1/predictive-maintenance-withsensorsinutilities
[66] K. Liulys, "Machine Learning Application in Predictive Maintenance," in *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream)*, Vilnius, Lithuania, Apr. 2019 - Apr. 2019, pp. 1–4.
[67] K. Kersting, "Machine Learning Applications, Ensemble Methoden," 2020.
[68] G. Drakos, *Support Vector Machine vs Logistic Regression.* [Online]. Available: https://medium.com/@george.drakos62/support-vector-machine-vs-logistic-regression-94cc2975433f (accessed: Apr. 6 2020).
[69] I. Steinwart and A. Christmann, *Support vector machines,* 1st ed. New York: Springer, 2008.
[70] L. Wang, *Support Vector Machines: Theory and Applications*. Berlin Heidelberg: Springer-Verlag GmbH, 2005.
[71] F. Pedregosa *et al.,* "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[72] A. Baru, "Three Ways to Estimate Remaining Useful Life for Predictive Maintenance," MathWorks Inc., 2019. Accessed: Apr. 9 2020. [Online]. Available: https://in.mathworks.com/company/newsletters/articles/three-ways-to-estimate-remaining-useful-life-for-predictive-maintenance.html
[73] X.-S. Si, Z.-X. Zhang, and C.-H. Hu, *Data-Driven Remaining Useful Life Prognosis Techniques: Stochastic Models, Methods and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017. [Online]. Available: https://ebookcentral.proquest.com/lib/gbv/detail.action?docID=4797402
[74] I. MathWorks, *Predictive Maintenance Toolbox.* [Online]. Available: https://www.mathworks.com/products/predictive-maintenance.html
[75] Siemens, *MindSphere.* [Online]. Available: https://siemens.mindsphere.io/en (accessed: Apr. 18 2020).
[76] Microsoft Azure, *Predicitive Maintenance with Ai.* [Online]. Available: https://github.com/Azure/AI-PredictiveMaintenance (accessed: Apr. 18 2020).
[77] Giovanni Di Orio, Gonçalo Cândido, and José Barata, "Self-Learning Production Systems: A New Production Paradigm," *InImpact: The Journal of Innovation Impact*, vol. 7, p. 887, 2014.
[78] R. S. Sutton and A. Barto, *Reinforcement learning: An introduction*. Cambridge, MA, London: The MIT Press, 2018.
[79] VDMA, Institut für Unternehmenskybernetik e.V., "Leitfaden Selbstlernende Produktionsprozesse," 2019.
[80] Hees, Dr. rer. nat. Frank, "Digitalisierungslösungen in der industriellen Anwendung," RWTH Aachen, Nov. 15 2018. Accessed: Mar. 17 2020. [Online]. Available: https://cybernetics-lab.de/uploads/2018-11-16_KI_IHK_-002.pdf
[81] R. Meyes *et al.,* "Motion Planning for Industrial Robots using Reinforcement Learning," *Procedia CIRP*, vol. 63, pp. 107–112, 2017, doi: 10.1016/j.procir.2017.03.095.
[82] P. Ennen, P. Bresenitz, R. Vossen, and F. Hees, "Learning Robust Manipulation Skills with Guided Policy Search via Generative Motor Reflexes," Sep. 2018. [Online]. Available: http://arxiv.org/pdf/1809.05714v2
[83] Sergey Levine and Pieter Abbeel, "Learning Neural Network Policies with Guided Policy Search under Unknown Dynamics," in *Neural Information Processing Systems (NIPS)*, 2014.

[84] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine, “Collective robot reinforcement learning with distributed asynchronous guided policy search,” in *IROS Vancouver 2017: IEEE/RSJ International Conference on Intelligent Robots and Systems, Vancouver, BC, Canada September 24-28, 2018 : conference digest*, Vancouver, BC, 2017, pp. 79–86.
[85] D. L. Poole and A. K. Mackworth, *Artificial Intelligence*. Cambridge: Cambridge University Press, 2010.
[86] Lilia Rejeb and Zahia Guessoum, “The Exploration-Exploitation Dilemma for Adaptive Agents,”
[87] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013, doi: 10.1177/0278364913495721.
[88] E. R. Davies, *Machine vision: Theory, algorithms, practicalities,* 3rd ed. Amsterdam, Boston: Elsevier, 2005. [Online]. Available: http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10169929
[89] James Furbush, *HOW TO START A DEEP LEARNING FACTORY AUTOMATION PROJECT IN FIVE STEPS.* [Online]. Available: https://www.cognex.com/blogs/deep-learning/five-steps-to-implement-deep-learning-projects (accessed: Jan. 24 2020).
[90] Brian Benoit, *WHAT IS THE DIFFERENCE BETWEEN AI, MACHINE LEARNING, AND DEEP LEARNING FOR INDUSTRIAL AUTOMATION INSPECTIONS?* [Online]. Available: https://www.cognex.com/blogs/deep-learning/ai-versus-deep-learning-versus-machine-learning-in-industrial-automation (accessed: Jan. 24 2020).
[91] G. Dreyfus, *Neural Networks: Methodology and Applications*. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2005. [Online]. Available: http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10143854
[92] J. Johnson and A. Karpathy, *CS231n Neural Networks Part 1: Setting up the Architecture.* [Online]. Available: http://cs231n.github.io/neural-networks-1/ (accessed: Mar. 5 2020).
[93] M. A. Nielsen, *Neural Networks and Deep Learning*: Determination Press, 2015.
[94] J. Johnson and A. Karpathy, *CS231n Convolutional Neural Networks: Architectures, Convolution / Pooling Layers.* [Online]. Available: http://cs231n.github.io/convolutional-networks/ (accessed: Mar. 4 2020).
[95] V. Nigam, *Understanding Neural Networks. From neuron to RNN, CNN, and Deep Learning.* [Online]. Available: https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90 (accessed: Mar. 4 2020).
[96] MathWorks Inc., *Deep Learning with Matlab.* [Online]. Available: https://in.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html (accessed: Mar. 5 2020).
[97] Bundesministerium für Wirtschaft und Energie (BMWi) Indutrie $.0, “Künstliche Intelligenz (KI) in Sicherheitsaspekten der Industrie 4.0,” 2019.
[98] G. Cooper, *New Vision Technologies For Real-World Applications.* [Online]. Available: https://semiengineering.com/new-vision-technologies-for-real-world-applications/ (accessed: Mar. 30 2020).
[99] J. Hui, *Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and YOLOv3).* [Online]. Available: https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359 (accessed: Mar. 5 2020).
[100] J. Huang *et al.,* “Speed/accuracy trade-offs for modern convolutional object detectors,” Nov. 2016. [Online]. Available: http://arxiv.org/pdf/1611.10012v3
[101] Common Objects in Context, *Detection Evaluation.* [Online]. Available: http://cocodataset.org/ (accessed: Mar. 30 2020).

[102] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The Pascal Visual Object Classes (VOC) Challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.
[103] EdjeElectronics, *How To Train an Object Detection Classifier for Multiple Objects Using TensorFlow (GPU) on Windows 10.* [Online]. Available: https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10 (accessed: Mar. 5 2020).
[104] *Entwicklungsmethodik für mechatronische Systeme*, 2004.
[105] IBM, “IBM SPSS Modeler CRISP-DM Guide,” 2016.
[106] D. E. O'Leary, *Enterprise Resource Planning Systems*: Cambridge University Press, 2012.
[107] F. Peschke, *Product Lifecycle Management (PLM)*. München: Carl Hanser Verlag GmbH & Co. KG, 2017.
[108] J. Kletti, *Manufacturing Execution Systems - MES*. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2007. [Online]. Available: http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10172854
[109] micropede, *Shop.* [Online]. Available: https://micropede.de/shop (accessed: Mar. 13 2020).
[110] Tinkerforge, *Tinkerforge Products.* [Online]. Available: https://www.tinkerforge.com/en/ (accessed: Mar. 6 2020).
[111] J. F. Puget, *The Most Popular Language For Machine Learning Is …* [Online]. Available: https://www.ibm.com/developerworks/community/blogs/jfp/entry/What_Language_Is_Best_For_Machine_Learning_And_Data_Science?lang=en (accessed: Mar. 22 2020).
[112] OpenCV, *OpenCV.* [Online]. Available: https://opencv.org/about/
[113] Travis Oliphant, *NumPy: A guide to NumPy.* [Online]. Available: http://www.numpy.org/
[114] J. D. Hunter, “Matplotlib: A 2D Graphics Environment,” *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, 2007, doi: 10.1109/MCSE.2007.55.
[115] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, 27:1-27:27, 2011.
[116] Open AI, *Gym.* [Online]. Available: https://gym.openai.com/ (accessed: Feb. 14 2020).
[117] Endurosat, *CUBESAT PROTOBOARD.* [Online]. Available: https://www.endurosat.com/cubesat-store/cubesat-obc/protoboard/ (accessed: Apr. 10 2020).
[118] Google, *Tensorflow detection model zoo.* [Online]. Available: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md (accessed: Mar. 6 2020).
[119] tzutalin, *LabelImg.* [Online]. Available: https://github.com/tzutalin/labelImg
[120] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” Nov. 2013. [Online]. Available: http://arxiv.org/pdf/1311.2524v5
[121] R. Girshick, “Fast R-CNN,” 2015. [Online]. Available: https://arxiv.org/pdf/1504.08083
[122] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” 2015. [Online]. Available: https://arxiv.org/pdf/1506.01497
[123] J. Johnson and A. Karpathy, “Lecture 3, Loss Functions and Optimization,” Stanford University, 2017. Accessed: Apr. 20 2020. [Online]. Available: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture3.pdf
[124] J. Canny, “A Computational Approach to Edge Detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-8, no. 6, pp. 679–698, 1986, doi: 10.1109/TPAMI.1986.4767851.
[125] K. Sarkar, K. Varanasi, and D. Stricker, “Trained 3D Models for CNN based Object Recognition,” in *VISAPP: Porto, Portugal, February 27-March 1, 2017*, Porto, Portugal, 2017, pp. 130–137.

Appendix

The full code of the mentioned systems can be found on the gitlab link:

https://git.rwth-aachen.de/vk/sf40_robot