Joseph Thomas Thekkekara, BSc

Development of an autonomous assembly station and the provision of the relevant data for the AI

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Production Science and Management

submitted to

Graz University of Technology

Supervisor

Head: Univ.-Prof. Dipl-Ing. Dr.techn. Franz Haas

Institute for Production Engineering

Graz, March 2021

Foreword and acknowledgements

This thesis is written as completion to the Master's degree programme: Production Science & Management, at the Graz University of Technology. The master programme focuses on knowledge of production science, production technology and production Management. The subject of this thesis, Development of an autonomous assembly station and the provision of the relevant data for the AI, falls within the scope of the master's field of digitalization of production systems and Industry 4.0.

The topic is selected in co-operation with the Institute for Software Technology at the Graz University of Technology. The main goal is to build an autonomous industrial robotic system that can solve any assembly task with minimum human inputs. The mechanical and production aspects of the topic was linked to my research and the thesis. The topic has been very interesting and intriguing and I have been able to achieve a result I am very satisfied with.

I would like to express my gratitude to the following people:
Univ.-Prof. Dipl.-Ing. Dr.techn. Franz Haas,
Assoc.-Prof. Dr. Gerald Steinbauer,
Dipl.-Ing. Philipp Simon Eisele, BSc
for their supervision, encouragement and outstanding support.

Alex Maestrini, BSc
for his thoughts and help along the way.

And everyone not named, who supported me in the process.

Abstract

The World Robot Summit 2018 was held in Japan which provided a venue for high-level robot technology from around the world. The main aim of the summit was to serve as a setting for various competitions that helps to speed up technology development, promote social implementation and demonstrate robotic problem-solving abilities. One of the hardest challenges in the WRS was the Assembly Challenge (Industrial Robotic Category).

This thesis mainly aims at solving the assembly challenge put forth by the WRS 2018. The intent is to build an automation system that can achieve "level 5" production automation, i.e. zero change over time for new product introduction. The primary focus of the thesis is to provide relevant input data for the autonomous assembly system to generate an assembly plan which can build any given assembly or subassembly in the least possible time.

The approach presented is to derive a complete set of assembly information regarding the product or the assembly and enable the robotic system to reason an optimal sequence plan to build the final assembly. Finally, the developed system is tested with different specific corner cases which involve at least five different industrial applications and a wide variety of assembly elements.

List of acronyms

2D	2 Dimensional
3D	3 Dimensional
AC	Air Conditioning
AI	Artificial Intelligence
API	Application Programming Interface
APR	Assembly Precedence Relations
BREP	Boundary Representation
CAD	Computer Aided Design
DOF	Degree of Freedom
GCA	Geometric Constraint Analysis
GPS	Global Positioning System
IAM	Inventor Assembly
IGES	Initial Graphics Exchange Specification
IPT	Inventor Part
ISO	International Organization for Standardization
NC	Numerical Control
OGA	Ordering Genetic Algorithm
OWL	Web Ontology Language
RDF	Resource Description Framework
ROS	Robotic Operating System
STEP	Standard for the Exchange of Product Data
WRS	World Robotic Summit
XML	Extensible Markup Language

Contents

List of Figures

List of Tables

1 Introduction

This chapter provides insight into the background and problem description for this thesis, along with the aim, objectives, approach, focus and limitations. Since the thesis is part of a broader research work involving several other disciplines; it is necessary to first define the scope of the thesis to fully understand the problem and make necessary simplifications.

1.1 Background

The World Robot Summit is a platform designed to bring forth innovation in robotics and foster state-of-the-art autonomous manufacturing systems. The summit consists of the following four challenges.

- Industrial Robotics Category
- Service Robotics Category
- Disaster Robotics Category
- Junior Category

The assembly challenge is one of the subcategories of the Industrial robotics challenge. The challenge includes the following tasks, such as bin picking, kitting task and assembly task. All these tasks are designed in such a way that it includes 2D assembly, 3D assembly and a flexible assembly to be completed by an industrial robotic system.

The two main objectives of the assembly challenge are to maximize *agility* and *leanness*. Agility is defined as the time required by a robotic system to change from one product to another, whereas leanness is defined as the rate of reusability of the equipment to produce the new product.[1]

The levels of automation in next-generation production systems are classified as shown in Table 1.1. The levels range from 1 to 5; where level 1 automation is when no changeover is possible for a new product to level 5 automation which aims at zero changeover time for a new product. The main aim of the assembly challenge is to achieve " level 5 " automation in next-generation production

[1] Conf. Yokokohji et al. (2019), p. 876-99.

Table 1.1: Levels for the next-generation production system, Source: Yokokohji et al. (2019), p. 877.

	Aspects during setup change		Aspects during operation
	Agility	Leanness	Utilization rate improvement
Level 5	0 day for new product (Changeover on the same day)	100% continual use (Introduction of universal hands that are able to perform jig-less assembly for multiple products, etc.)	Machine learning (Temporal stoppage prevention/cycle time improvement)
Level 4	2 days for new product (Changeover on a weekend or an overnight business trip)	Available for new products only by recombining existing equipment. (Universal hands able to grasp multiple products, etc.)	Automatic recovery from temporal stoppage (Learning through observing human intervention, etc.)
Level 3	1 week for new product (Changeover in a week, e.g. during large consecutive national holidays)	50% or more can be reused. (Utilization of specialized hand library, flexible jig, multi arms, etc.)	Operation rate improvements (Prevention measures against temporal stoppages, etc.)
Level 2	1 month for new product	Reusing only robots	Reduction of temporal stoppage rate by absorbing part variations using sensors
Level 1	For specific products only (Changeover is not assumed.)	0% (No reuse is assumed.)	Controls parts variations to ensure an enough utilization rate. Human intervention is required for temporal stoppages

system, i.e., to build an agile and lean robotic assembly system which is possible to achieve zero change over time.[2]

The thesis expands on the assembly challenge laid out by the WRS to build an efficient manufacturing system that can achieve "level 5" automation in the most effective way.

1.2 Problem description

An Industrial assembly task can have different degrees of automation, depending on a variety of factors, including production lifetime, volume, and flexibility. To achieve the "level 5" automation as described in the previous section; it was decided to choose the assembly model outlined by the assembly challenge (Industrial Robotics Category) presented at the WRS 2018. The test setup consists of a belt drive unit, as illustrated in Figure 1.1, to be autonomously assembled without any human interference.

[2] *Ibid.*, p. 876-99.

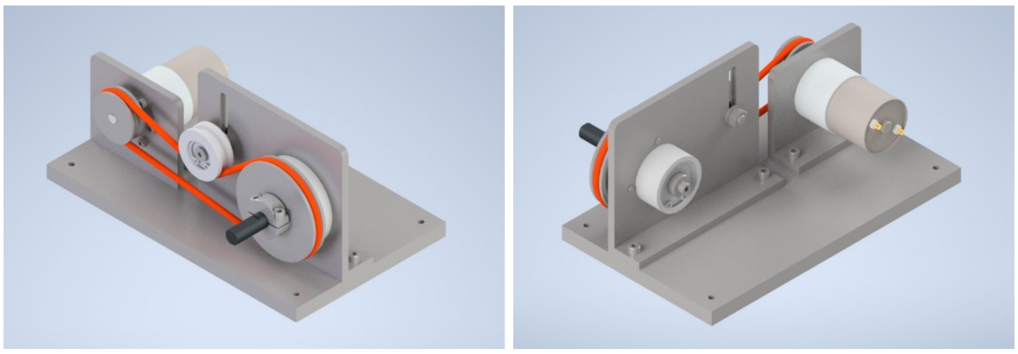


Figure 1.1: The belt drive unit used for the assembly challenge at the WRS 2018, Source: Rendering from own assembly drawing, based on the industrial robotics assembly challenge rulebook (2020), p. 44-69.

1.3 Aim and objectives

The thesis aims to solve three central challenges to successfully perform the autonomous assembly.

1. Identify the components and subassemblies that makes up the final assembly or product.
2. Derive an assembly plan to assemble the final end product.
3. Execute the derived plan in a real-world robotic system.

1.3.1 Identify the components and subassemblies

First, the system needs to be able to identify the basic building blocks of the presented assembly. Every assembly is constituted of components and subassemblies. The system should be able to geometrically and semantically understand the constituent components generated by the design engineers in the form of CAD models. The computer aided model provides knowledge about shape, size, material properties, functions, and assembly relation of components with one another.

1.3.2 Derive an assembly plan

Second, the system needs to be able to derive an assembly plan from the knowledge and descriptions received. Here the system needs to plan on the abstract as well as the continuous level.

Three different things need to be planned for

- part movements
- tool movements
- the reasoning for the assembly steps to build the product

The assembly task needs to be automatically divided into sub-tasks. Every sub-task can either be a basic robotic skill or again subdivided into relatively complex sub-tasks or a combination of robotic skills. Each step should be based on combinations of already pre-acquired skills. In the modern flexible automation systems, these skills are usually acquired through tactile learning or using complex machine learning algorithms.

The proposed system should be able to reuse and combine these multiple skills flexibly by referring to the component models and their attached reference frames. The idea here is to minimize manual intervention by using derived knowledge of every single components used in the assembly task. The aim is to have as little information about the final product, intermediate product, or any assembly sequences as possible from the user.

1.3.3 Execute the derived plan in a real-world robotic system

Third, the proposed system needs to be able to execute the assembly plan in a real use case autonomously and robustly. This requires that the system can identify and precisely locate the involved parts and tools using sensor, vision system and perception system. Moreover, the system needs to complete the planned skills based on the actual configuration in the environment. To gain robustness, the robotic system needs follow up capabilities to track the progress of skills and assembly steps and be able to re-plan assembly steps on the fly.

1.4 Approach and methods

The following part describes the assignment derived from the aims and objectives described in Section 1.3. The definitions already referred to above gives a very clear picture of what the outcome should be:

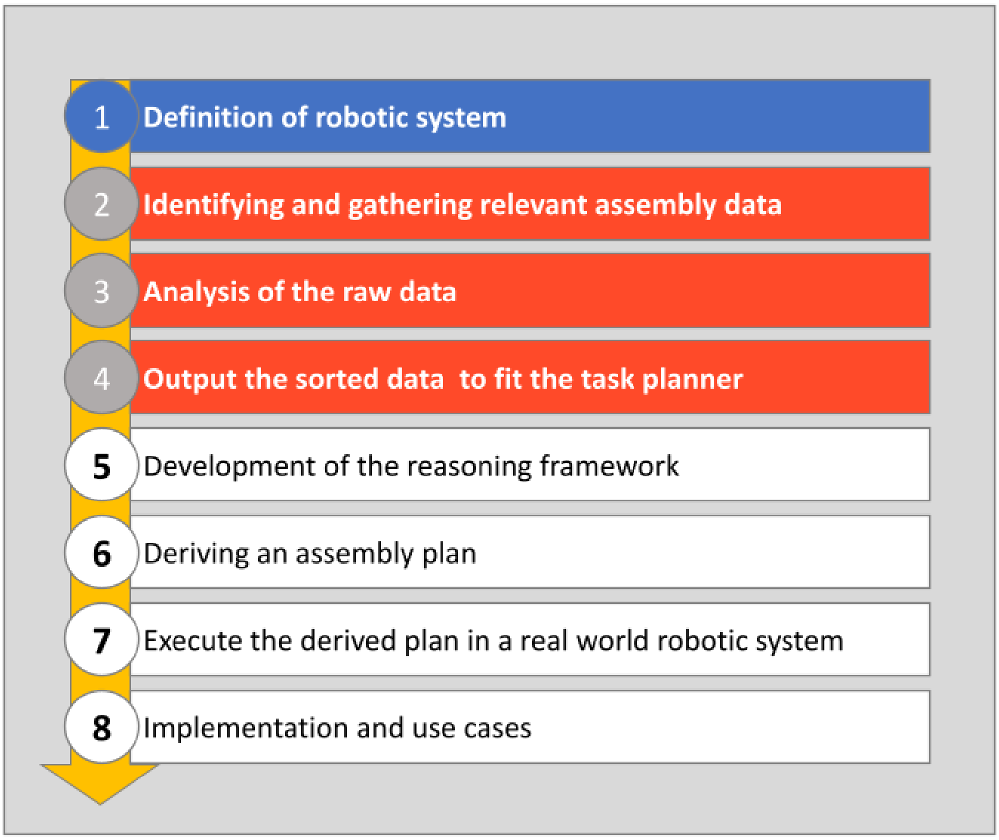

1 Definition of robotic system
2 Identifying and gathering relevant assembly data
3 Analysis of the raw data
4 Output the sorted data to fit the task planner
5 Development of the reasoning framework
6 Deriving an assembly plan
7 Execute the derived plan in a real world robotic system
8 Implementation and use cases

Figure 1.2: Approach for the thesis, Source: Own illustration.

- To identify the relevant assembly data to be provided to the robotic architecture.
- To have a system that will extract and analyse all the necessary data from the CAD Model.
- To provide the task planner with the input data necessary to complete the autonomous assembly process.

In the beginning, the theoretical framework for the state of the art manufacturing systems that involves industrial robots is described. The robotic system must be defined along with its various constituting components. Literature research about relevant aspects of the autonomous system has been carried out. The literature research investigates the various researches published on autonomous assembly system, flexible production, and sequence planning.

The second part of the thesis deals with the method as described in steps 2, 3 and 4 of Figure 1.2. The relevant assembly data gets first identified and gathered. It is then followed by analysing the data which gets fed into the task planner. The output of the proposed system is what drives the robot framework. It always must be taken into consideration how the CAD system interacts with the software part of the robotic subsystem and describes the interface between the *production* and *software* aspects of the thesis.

The final phase of the thesis deals with the implementation of the collected data as described in steps 5, 6, 7 and 8 of Figure 1.2. This phase begins with the development of the reasoning framework. The framework then derives the necessary assembly plan. The robotic system puts together the assembly or subassembly using the generated assembly plan.

1.5 Focus

This work focuses on “level 5” automation, the highest degree of automation, to model a system that autonomously builds an assembly plan to completion. The idea is to automate the assembly system to becomes an autonomous intelligent system. The system should be able to derive an assembly plan from a descript-ion of the product automatically and execute this assembly plan autonomously. “Level 5” automation, in comparison to fully pre-programmable automation, has the advantage of being able to detect product changes or even completely novel products on its own and can react to change with no or a minimum of delay time, adaptations or manual labour.

1.6 Limitations

The main limitations for such a system are to include all the use cases that come under the broad umbrella of all the assembly operations that are performed on a production line. The scope of this thesis is limited to assembly processes that involve pick-and-place, insertion, and screwing operations.[3] There are a lot more use cases like joining, welding, and riveting that have not been considered in regard to the thesis.

[3] Conf. Fujita et al. (2019), p. 1-15.

2 Theoretical framework

In this chapter, the related theoretical framework is presented to attain a more profound understanding of the subjects related to this thesis.

The theoretical references are divided into three main topics, Industrial Assembly Workflow, Essential elements of Robotics, and Assembly Model Data, which will be executed throughout this thesis.

2.1 Industrial assembly workflow

Assembly workflow in a production or assembly line should be clearly understood before the system can be converted into an autonomous system. There are three different kinds of assembly line configurations.[4]

- manual assembly line
- collaborative assembly line
- programmable/autonomous assembly line

The *manual assembly line* constitutes majorly of human assembly workers who follow an assembly plan to produce a product as illustrated in Figure 2.1 (a).[5] The necessary components needed for each step of the assembly process is usually provided as a "kit" which facilitate a more efficient assembly workflow.[6] The kit can come in a variety of different forms like a bag, a tray, compartments, or a kitting tray with compartments and part holders. The kitting trays minimizes the preparation time for the assembly and bring in a flow to the assembly process.

The *collaborative assembly line* constitutes a mix of human and collaborative robots who work together to produce an end product. This kind of an assembly line is much more efficient in terms of flexibility and dealing with complicated assembly tasks.[7]

[4] Conf. Yokokohji et al., *supra* note 1, p. 876-99.

[5] *Ibid.*, p. 876-99.

[6] Conf. Drigalski et al. (2020), p. 1-17.

[7] Conf. Yokokohji et al., *supra* note 1, p. 876-99

The *programmable/autonomous assembly line* consists of industrial robots that work together to produce an end product as illustrated in Figure 2.1 (b).[8] In the current industry scenario, these robots are usually programmed to fit the product requirement and reprogrammed when there is a product change. In the thesis, this system is expanded to include complete automation of the assembly process by the Industrial robots. The autonomous robots handle the whole assembly workflow from the start of picking individual parts to a completely assembled product.

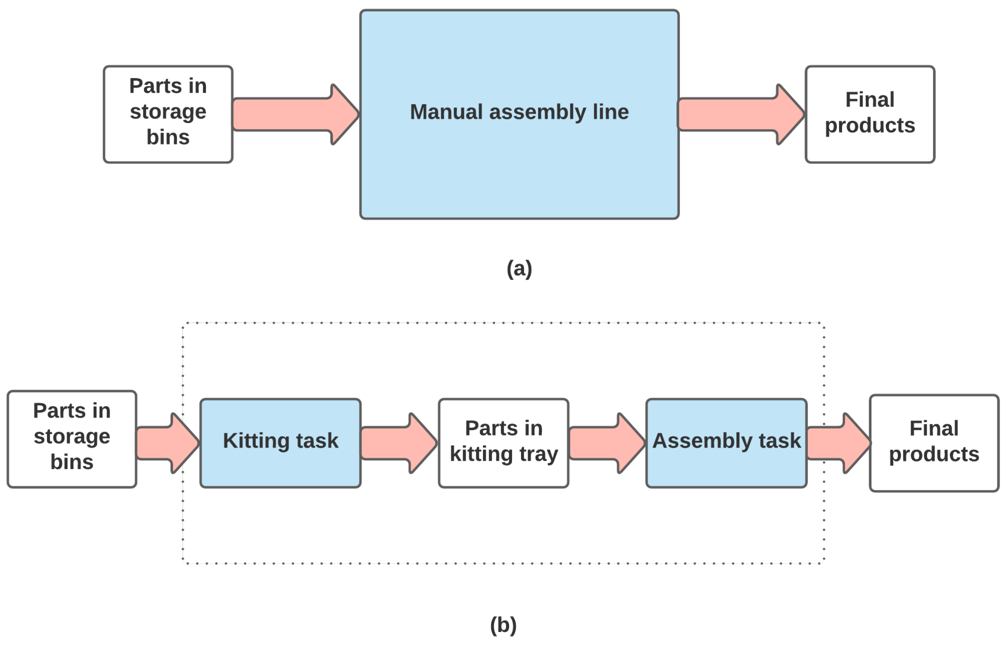


Figure 2.1: Comparison of assembly workflow in different production environments. (a) Manual assembly line. (b) Assembly processes in autonomous assembly system, Source: Own illustration, based on Yokokohji et al. (2019), p. 880. (modified).

2.1.1 Assembly tasks at the WRS 2018

This section, describes all the different individual tasks associated with the Assembly challenge at the World Robotic Summit.[9] It has three assembly tasks with the increasing order of difficulty: a 2D assembly task or task board, a 3D assembly task, and a 3D assembly task with a flexible assembly. The assembly challenge is designed in such a way to simulate the typical industrial workflow in a production environment as discussed in Section 2.1.

[8] *Ibid.*, p. 876-99.

[9] Conf. WRS (2020), online source [27.09.2020].

Kitting task

Kitting is defined as the assembly task in which the parts are picked up from the part bins or component trays and assembled into a kitting tray which comprises accurate part positions in predefined poses. The assembly kits can either be used in a manual assembly line or an autonomous assembly process depending on the use case. A skill-based architecture can be used for placing parts in an industrial kitting scene. The architecture is capable of accurately placing objects in confined areas and narrow compartments.[10]

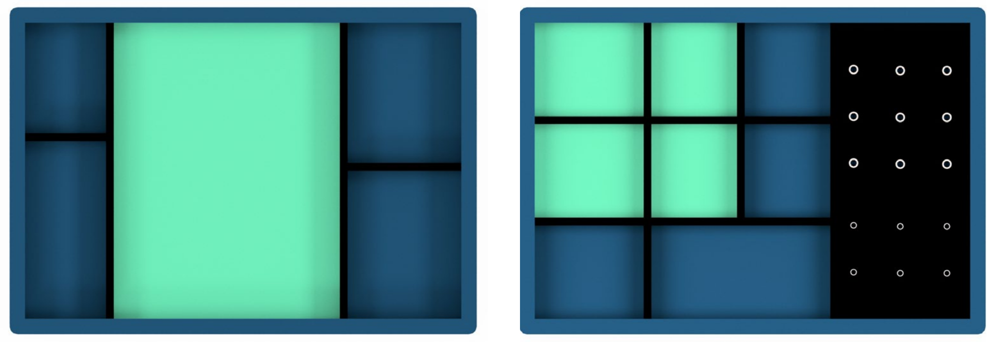


Figure 2.2: Kitting tray, Source: Rendering from own assembly drawing, based on the industrial robotics assembly challenge rulebook (2020), p. 12-14.

The kitting tray shown in Figure 2.2 is used in the WRS challenge and assembled by a robotic system. The kitting task is used in this case to simulate the ”bin picking” scenario used in the assembly process.

Assembly task

The Assembly challenge is used to simulate the various 3D assembly operations performed by the robotic system in an industrial setting. The challenge involves the complete assembly of a belt drive unit which consists of multiple sub-assemblies. This belt drive unit is a scaled-down version of a mechatronic component that is usually used inside various home automation systems, computer accessories or AC systems.[11] A completely assembled kitting tray is used as shown in Figure 2.2 as the base and follow the workflow as shown in Figure 2.1 (b).

[10] Conf. Polydoros et al. (2016), p. 255–268.

[11] Conf. Drigalski et al., *supra* note 6, p. 1-17.

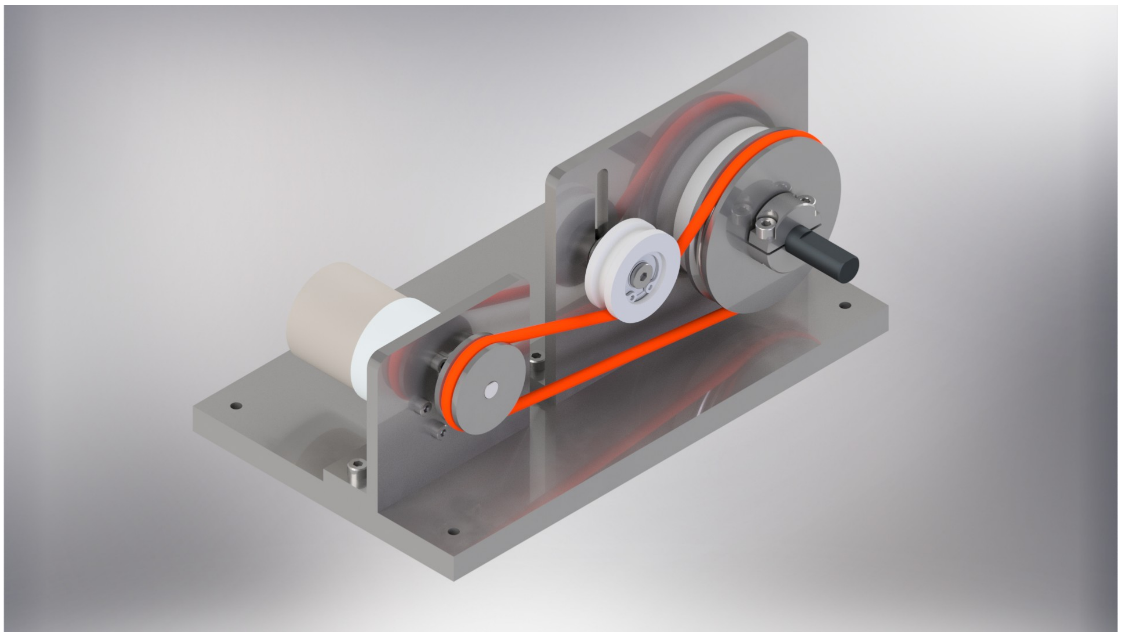


Figure 2.3: The belt drive unit with flexible part, Source: Rendering from own assembly drawing, based on the industrial robotics assembly challenge rulebook (2020), p. 44-69.

Flexible assembly task

The flexible assembly task is an extension of the assembly task and involves a more complex subassembly that gets put together.[12] New and different assembly parts are introduced into the existing system, and the ability of the robotic system to adapt to the newly introduced part is measured. The manipulations strategies for the new parts inside the flexible assembly are completely different from those of the parts that are already known, making it a "level 5", automation task. The aim is to achieve zero change over times and the robotics system must prove its flexibility in switching between new components and parts.

2.2 Industrial robots and classification

A robot is defined as a device that (a) performs programmed operations or operates by remote control, (b) senses external conditions and events, and (c) uses feedback derived from ongoing events and operations to modify its own

[12] Conf. Drigalski et al. (2019), p. 1-14.

actions in accordance with the data it senses.[13] An industrial robot is an automatically controlled, reprogrammable, multipurpose manipulator programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation applications.[14]

Robots can be classified based on different parameters like Degrees of Freedom, robotic motion, the type of platform used, power source, application areas or intelligence.[15] Based on the degrees of freedom industrial robots are classified into the following four categories.

- Articulated robots
- SCARA robots
- Delta robots
- Cartesian robots

2.2.1 Articulated robots

An articulated robot has the highest degrees of freedom compared to all the other robots. They are usually classified by the number of points of rotation or axes they have. [16]

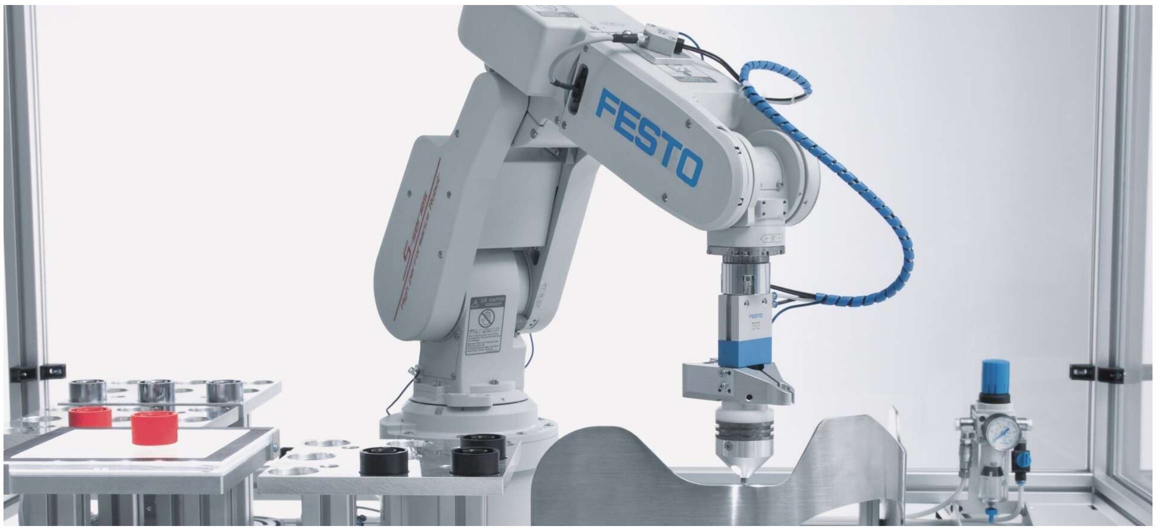


Figure 2.4: Articulated robot, online source [27.09.2020].

The advantages of articulated robots are that they can perform tasks that span across non-parallel planes. They usually have a more complex kinematics and

[13] Conf. Weik (2001), p. 1499.

[14] Conf. ISO 8373 (2012), online source [27.09.2020].

[15] Conf. Poole (2012).

[16] Conf. Matt Minner (2019), online source [27.09.2020].

a relative higher component mass compared to the other similar robot categories.

The main application areas of articulated robots are as follows [17]

- Pick and place
- Assembly
- Material removal
- Machine tending
- Palletizing

2.2.2 SCARA robots

Selective Compliance Assembly Robot Arm (SCARA) combines inflexible motion in the vertical plane with compliance or flexible positioning in the horizontal plane allowing faster part insertions during assembly tasks.[18] These robots have been specifically designed and built for light applications like pick and place and small part handling. They can achieve very high cycle times with high accuracy and can be integrated easily into the assembly lines.

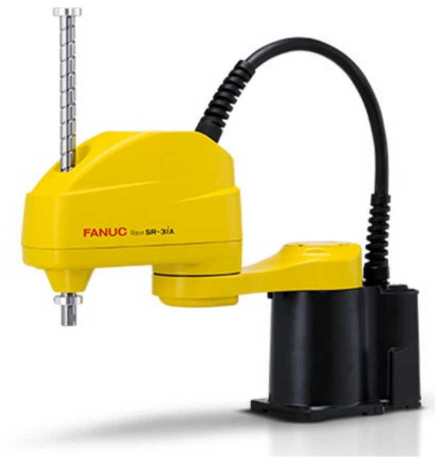


Figure 2.5: Fanuc SR31A SCARA Robot, online source [27.09.2020].

The main application areas of SCARA robots are as follows [19]

- Pick and place
- Assembly
- Packaging
- Inspection
- Dispensing

[17] *Ibid.*

[18] Conf. International Federation of Robotics, online source [27.09.2020].

[19] Conf. Matt Minner, *supra* note 16.

2.2.3 Delta robots

Delta robots also referred to as parallel robots are usually high-speed pick and place robots that are usually placed above the working zone.[20] They are normally used in conjunction with a vision system to pick pieces randomly placed in complex sorting and packaging applications. The delta robot's payload capacity is generally much lower than alternative technologies. Longer reaches is a limitation for this kind of robots.

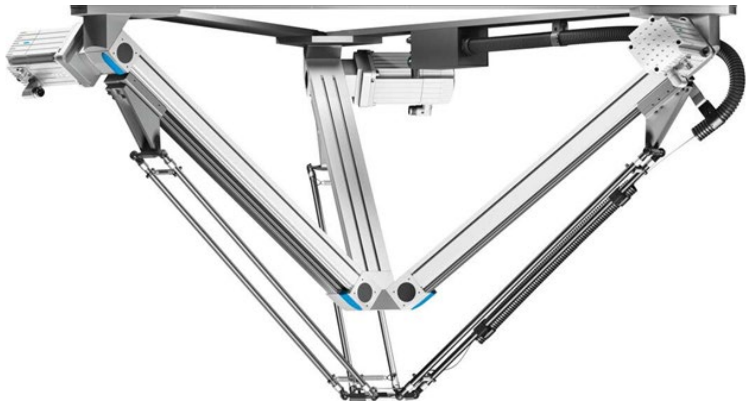


Figure 2.6: Parallel kinematic delta robot, online source [27.09.2020]

The main application areas of delta robots are as follows [21]

- Pick and place
- Assembly
- Inspection

2.2.4 Cartesian robots

Cartesian robots consist of three linear joints that make sure that the end-effector of the robot always moves in the Cartesian coordinate.[22] The robotic design consist of an assembly of linear actuators and sometimes a rotary actuator at the end of its arm for 3D applications.

Examples of this type of robots are gantry robots or robots that are used for stacking parts in bins. The attached end effectors in these robots can allow for rotational movement.

[20] Conf. Festo (2020), online source [27.09.2020].

[21] Conf. Matt Minner, *supra* note 16.

[22] Conf. International Federation of Robotics, *supra* note 18.

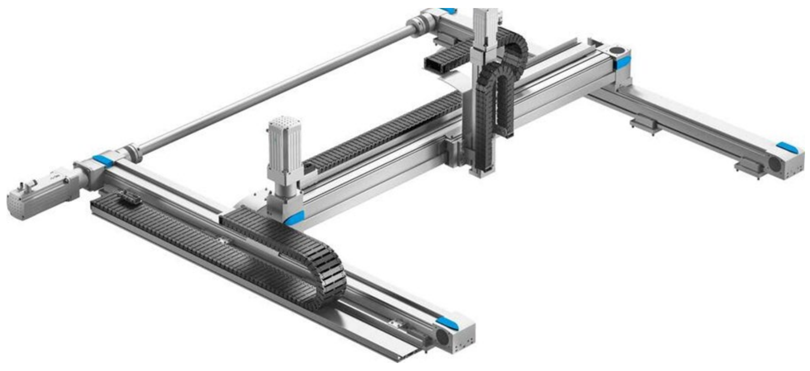


Figure 2.7: Three-dimensional gantry robot, online source [27.09.2020]

The main application areas of delta robots are as follows [23]

- Pick and place
- Assembly
- Inspection
- Dispensing

2.3 Essential elements of robotics

In this section, the essential elements of robotics are presented. The definition of a robotic system is first defined and the different constituent elements that makes up the robot and the robotic environment is defined.

For the robot to reach its end state it must go through the following sequence:

- Perception
- Decision making
- Action

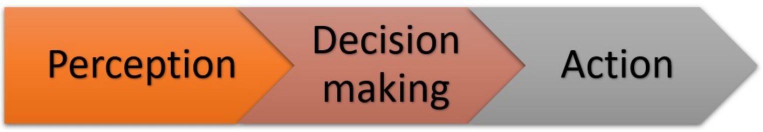


Figure 2.8: The sequence of steps that a robotic system undergoes to perform an action, Source: own illustration.

[23] Conf. Matt Minner, *supra* note 16.

2.3.1 Perception

The perception system of a robotic system is the sensory system that assists the robot to recognise and identify the surroundings. This can include different kind of perception systems like cameras, lidar, radar or ultrasound to perceive vision. Temperature and pressure sensors to perceive touch. The two main types of perception systems are described below: the vision system and the sensor system.[24]

2.3.1.1 Vision system

The vision system is the part of the robotic system which can detect and identify the various components or parts, obstacles and all the physical objects that are placed on the worktable or in the surroundings of a robot. There are different kinds of vision systems that are available in the market such as 2D cameras, lidar, radar or other applied sensor systems.[25]

In the described case, cameras can be used for object detection, to identify the position of different parts, or further used to check the positions of the kitting trays on which the different parts are placed and the orientation of the different parts inside the kitting tray.

2.3.1.2 Sensors

The robotic system employs different kinds of sensor systems to deal with different scenarios. For the robot to understand the various uncertainties in the part placement and grasp misalignments, various feedback sensors are used. Force-torque sensors are normally used for object insertion tasks. Force sensors can also assist in identifying parts or ensuring exact positioning.

Sensors provide the robotic system with the necessary information to perform all the required operations. Sensors provide the robot with information about the robot’s action (internal sensors) or about the external environment (external sensors). The robotic system uses two different classes of sensors. The two categories are as follows:

- Internal state sensor
- External state sensor

[24] Conf. Udacity (2020), online source [27.09.2020].

[25] Conf. Tajima et al. (2019), p. 1-15.

Internal sensors

Internal sensors perform internal state sensing in the robotic system. The sensors notify the robot of the position, velocity, and acceleration of its joints. It gives the robot the ability to sense where the different parts of the kinematic chain of the robotic arm are in a spatial system. Robotic systems mainly use different classes of internal sensors to describe position, velocity, force, and acceleration. [26]

Position measurement

When it comes to position measurements, translational as well as rotational movements should be measured. There are two main types of movement sensors that are used when it comes to measuring translational motion. Either using a linear potentiometer or using a magnetic measurement method.

For measuring the rotational movements, many different devices are available. For example, an optical encoder can provide incremental or absolute rotational measurements.[27] A resolver consisting of a movable magnetic coil that rotates inside several fixed coils is another angular position measurement device.

Velocity measurement

In robotic applications, the speed of the arm movement must be measured. A tachometer is the most common speed measurement device. Tachometers are devices that produce an output voltage proportional to their speeds.[28] Hall effect generators are also used to sense the speed of a rotating shaft.

Force measurement

Force measurement is used to detect the amount of load on a robotic arm. The weight of the object on a robot's arm can help determine the potential overload condition on the robot. Strain gauges are widely used devices for internal force sensing applications. [29]

Accelerometers

Acceleration can be calculated by measuring the change of speed over a given period of time. Strain gauges can be used to measure the force created by the

[26] Poole, *supra* note 15.

[27] Conf. Kojima et al. (2004), p. 493-98.

[28] Conf. Yoshikawa (1990).

[29] Poole, *supra* note 15.

acceleration of a known mass. It is calculated using newtons law of motion. Accelerometers can be used as joint position sensors in robotic manipulators.

External sensors

External sensors provide the robot with the necessary information about the surroundings of a robotic environment and information about the object that is to be manipulated. External sensors vary from contact sensors to sensors measuring temperature and movements. [30]

Pressure transducers

Pressure transducers detect the external pressure or force on an object. Pressure transducers can be used to deduce the shape and orientation of an object. The pressure transducers help in determining the amount of force required to close the grippers to hold an object in place.

There are mainly three types of sensors available to measure the pressure exerted by an end effector on an object: conductive elastomer sensors, optical sensors, and silicon strain gauges. There are also other types of pressure sensing devices like carbon fibres, conductive rubber, and elastomer sheets. [31]

Motion detection

Motion detection or slip sensing devices detect the movement of objects due to gravity or some other external force. The motion detectors use feedback loops to tighten the gripper if slippage occurs. [32]

2.3.1.3 Progression of sensor data

There has been a significant change in the calculation of the relevant data from the sensor system. The transmission of raw sensor data has been complemented with the use of on-line programming where the position, orientation and other data are calculated and collected with the assistance of a teach pendant and by using collaborative robots. These robots can learn by using a human assistant who can teach each step of the program which can be later executed as a single program.

[30] *Ibid.*

[31] Conf. Almassri et al. (2015).

[32] Poole, *supra* note 15.

2.3.2 Decision making

The robotic system takes the inputs that are derived from the perception system and its internal state and then comes up with a decision to accomplish the given task. The software framework of the robotic system is responsible for the decision-making process.[33]

Software framework

The software system enables the robotic system to become autonomous. The Robotic Operating System (ROS) is a set of tools and libraries that can act as base for writing various robotic software.[34] To organise robotic knowledge and behaviour, a skill-based platform SkiROS, was developed on top of ROS.[35] Skill based assembly planning is based on primitives or tasks, that are a sequence of skills that are integrated into the robot control platform.

Robotic motion planners on the other hand help in generating the robotic motion of the arms to perform various skills or assembly tasks.[36] But in the case of an autonomous assembly system, the robot must plan the trajectories and path motion by itself.

The decision-making process in an assembly robot can range from simple decision trees to complex neural networks. The difference between a decision tree and a neural network is described below.

Decision tree

A Decision Tree is a directed tree that represents a list of nested yes-no statements that are used to derive decisions.[37] Each node in the decision tree can describe decisions, conclusions, or actions to be carried out.

Figure 2.5 shows a decision tree executing a task. The tree is generally traversed in a top-down manner.

[33] Conf. Udacity, *supra* note 24.
[34] Conf. Schlette et al. (2019), p. 1-17.
[35] Conf. Rovida et al. (2017), 121–160-121–160.
[36] Conf. Wang, Harada & Wan (2020), p. 1079-93.
[37] Conf. Colledanchise & Ögren (2018).

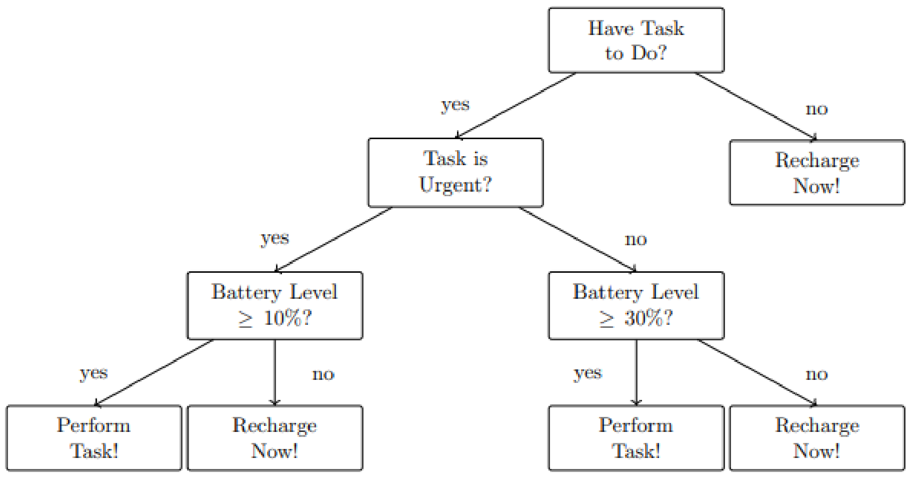


Figure 2.9: Example of a Decision Tree executing a generic robotic task, Source: Colledanchise & Ogren (2018), p. 36

In the context of the Master Thesis, a decision tree is used to analyse the parent child relationship between every single component inside the given assembly. The relationship constraints between different individual parts are used as the base to build up the parent-child relationship and the assembly plan.

Neural network

Neural networks are mathematical models that use learning algorithms or a network of functions to understand and translate a data input of one form into the desired output, usually in another form.[38]

[38] Conf. Keijsers (2010), p. 257-59.

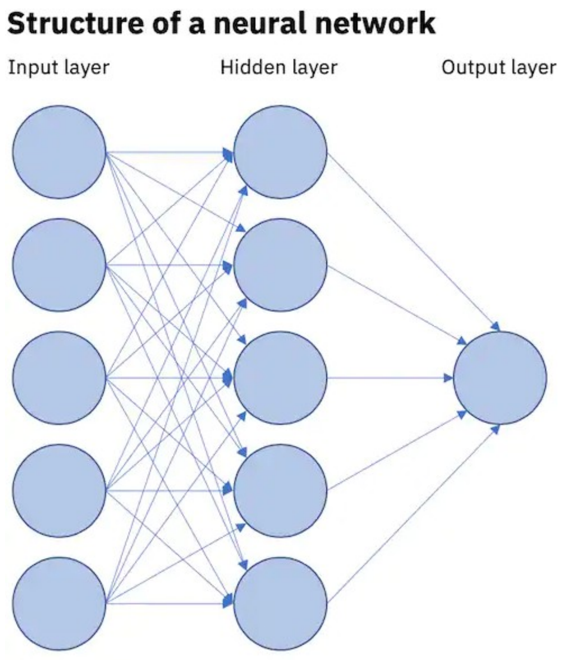


Figure 2.10: Structure of a neural network, Source: Eda Kavlakoglu (2020), online source [27.05.2020]

Most neural networks tend to flow in the forward direction as shown in Figure 2.6. The model flows from the input layer to the output layer passing through the hidden layer.

2.3.3 Action

The action is the end goal of the robotic system and it takes input from the perception system first. Then it uses the decision tree to arrive at a possible solution and finally pass on the requirement to perform the required action.[39]

For the robotic system to perform the various required actions, the following tools and manipulators are used.

Robotic manipulators

The most important component when it comes to the assembly system is the robotic manipulators.[40] They are mounted in an assembly line on either side and usually rests on vibration resistant structures. In the context of the thesis, the

[39] Conf. Udacity, *supra* note 24.

[40] Conf. Tennomi et al. (2020), p. 1-15.

system can scale to include different types of standard manipulator arms.

Screwing tools

The screwing tool is used to pick and place screws and fasten them with the required torque. The screwing makes up an important part of the assembly process and hence a really important element of the system. Different methods can be employed to perform a screwing operation; like grabbing a screw bit using a self-centering gripper, using the gripping mechanism to hold them in place or use a dedicated screwing tool.[41]

Tool changers

Tool changers consist of different categories of end effectors that can either be grippers or tools. Grippers themselves can be of different categories like a parallel gripper, three-fingered grippers, mechanical joints, suction cups or other pneumatic grippers. Grippers usually have a force sensor inside them which helps them to measure the various forces encountered while executing various skills.

In conventional assembly lines, jigs and fixtures were used for centering or aligning parts, but in case of autonomous assembly, we try to minimize their use so that zero change over time is possible without using any additional elements in the system.

2.4 Current approaches in robotic programming

This section describes the existing systems as well as the state-of-the-art systems in current robot programming. To propose possible improvements to the robotic system we need to first analyse and classify the existing programming approaches. The three most common approaches are classified below:

- Off-line programming
- On-line programming
- Digital twins

[41] Conf. Nie et al. (2020), p. 1-22.

Off-line programming of an industrial robot for manufacturing

Off-line programming is defined as the robot programming in which the mechanical robot and other robotic systems are not engaged during programming.[42] The programming is usually performed on a dedicated computer. The main advantage of off-line programming is that the robotic operations can be simulated even before the robotic system is built.

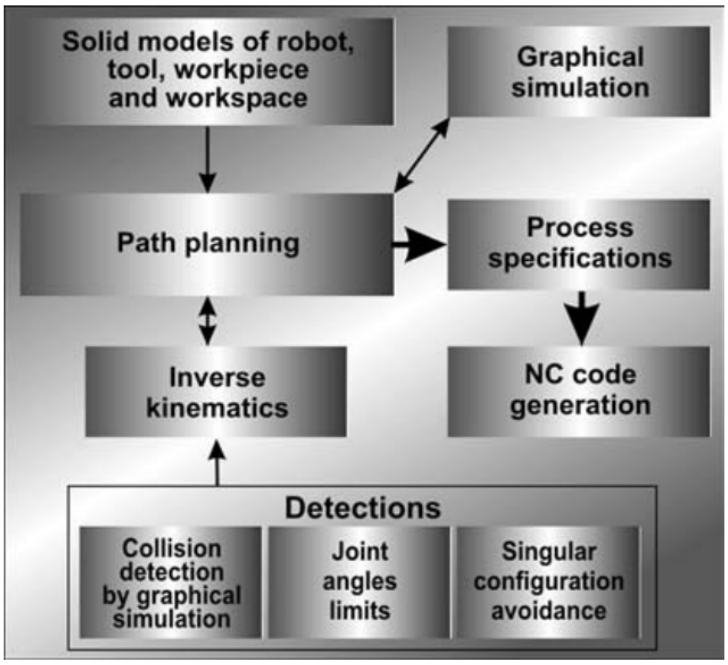

Solid models of robot, tool, workpiece and workspace
Graphical simulation
Path planning
Process specifications
Inverse kinematics
NC code generation
Detections
Collision detection by graphical simulation
Joint angles limits
Singular configuration avoidance

Figure 2.11: Automated robotic system architecture, Source: Mitsi (2005), online source [27.05.2020]

An off-line robotic system that generates the NC-code for a given manufacturing process is illustrated in Figure 2.11.[43] The off-line programming system consists of a graphical simulation of the robot and its environment, a kinematic model of the robot and path planning which generates the Numerical Control code for the manufacturing process. The robotic behaviour can be simulated using the method described above eliminating various planning problems like the collision, limits, timings, etc.

[42] Conf. Nilsson (1996), p. 21-45.

[43] Conf. Mitsi et al. (2005), p. 262-67.

On-line programming of an industrial robot for manufacturing

On-line programming is defined as the robot programming in which the mechanical robot is engaged during programming. In case of on-line programming the teach pendant is used to move the end effector to the required position and orientation. Each step of the programming task is repeated using this method to record a sequentially written robot program. [44]

One of the main advantages of online programming is simplicity. But complex motion along well-defined mathematical paths like turbine blades or aircraft wings are hard to achieve using on-line programming.

Digital twins

A digital twin is defined as the digital counterpart of a physical robotic system that can mirror all the actions performed by the physical system. The digital simulation can mirror the real-time operating conditions of a physical system.[45]

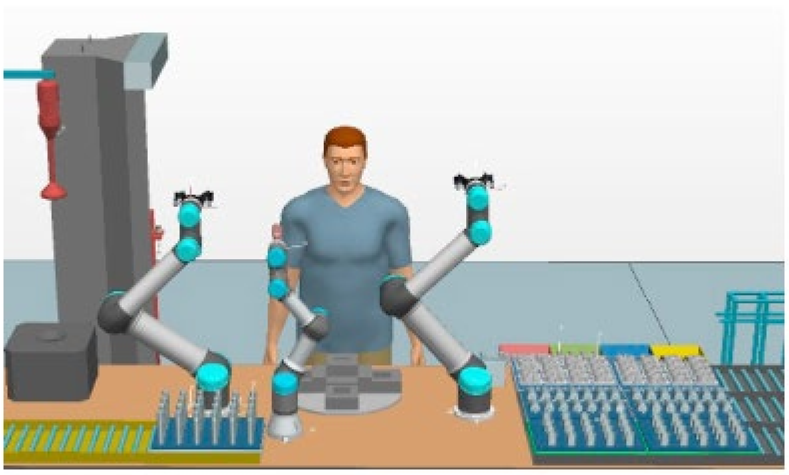


Figure 2.12: Digital twin framework of a human robot work cell: Malik & Bilberg (2018), online source [27.05.2020]

Figure 2.12 shows an example of a digital twin framework of a human robot work cell. The virtual model in the digital twin is build based on four different layers i.e, geometric layer comprising of the 3D CAD objects, the position and placement of the 3D objects in the modelled workspace, the interaction of the objects and the robotic system including physics and kinematics and finally the assembly sequence process.

[44] Nilsson, *supra* note 42.

[45] Conf. Malik & Bilberg (2018), p. 278-85.

An integrated programming approach

To include all complex use cases and ensure a flexible robotic manufacturing system, the best approach is to use a combination of all three programming methodologies. The robotic control system should be built to support autonomous assembly planning. A dedicated planning algorithm should also be written to ensure sequential assembly planning.

2.5 Assembly model data

A mechanical assembly is built up of modules that are either parts or subassemblies. A traditional assembly model comprises of information regarding parts, their relationships and various material properties.

The Computer Aided Design data of the assembly is the input that is necessary to automatically generate the assembly sequence to perform the autonomous assembly process. The design data contains the knowledge base about all the different attributes of an assembly.

The main goal of the thesis is to develop a system that can enable model change on the fly.[46] The developed system should efficiently extract all the available assembly model data to build the reasoning mechanism and the assembly planner. A flexible skill-based robotic system can be used to achieve the required autonomy. The skill-based robotic system should also be able to plan for flexible re-assembly.[47] An alternative approach to the re-assembly framework is the use of a disassembly sequence planning framework. A hierarchical approach is used for achieving the disassembly sequence planning.[48] The subassemblies are identified using a method that is based on collision information.

2.6 Simplification

One of the major simplifications that were made was the vision system. In our system structure, the scope of the vision system was too large. The vision system was simulated to a near-perfect perception system that can detect all parts at 100% accuracy. The robotic system is also simulated in a simulation environment instead of using the hardware manipulators for the assembly process. All parts for the assembly were also modelled inside the CAD system including three non-standard parts.

[46] Conf. Soetebier et al. (2008), p. 341-46.

[47] Conf. Gilday, Hughes & Iida (2018), p. 1-9.

[48] Conf. Ebinger et al. (2018), p. 3548-55.

2.7 Theoretical framework summary

This section summarizes the theoretical framework behind the thesis. Firstly, the different types of assembly lines and the production workflow necessary to complete the assembly of the belt drive unit are defined. The different sequence of assembly tasks that must be followed are then described. Finally, the system architecture is defined along with the necessary simplifications.

3 Method

The project methodology consists of three phases: the execution phase, implementation and validation and the case studies.

Phase one is the execution phase, which consists of the conceptual model, the assembly sequence planner, and system implementation. In phase two the developed model that is presented is validated and verified on the WRS 2018 belt drive unit. And in phase three the proposed model is tested to include as many different use cases as possible.

3.1 Conceptual model

The geometrical product specifications of any product are usually generated using a Computer Aided Design tool. This design is usually produced by the Engineering or design team of a company and later shared across to manufacturing. An overview of the Robot architecture is represented in Figure 4.1. The grey dotted squares represent the CAD node and the Robot node whereas the yellow rectangles represent the various plug-ins.

To make autonomous planning possible, the generated CAD product data needs to be fed into the task manager. The CAD system contains two different levels of information, Geometric Entities which represents the actual 3D model of the product or assembly and Geometric Constraints which represents the relationships between the different components of the product. The task manager is responsible for taking the generated CAD data that is described above and generate the assembly plan. It then dispatches the assembly plan to the skill manager which executes the various hardware-based primitives (grasping, placing, driving etc.) and skills (pick, locate etc.).

The scope of this thesis focuses on the CAD node represented in the Figure 3.1. The primary goal is to generate suitable output that must be inputted into the robot node. For developing the conceptual model of the autonomous assembly system, we start by analysing the literature review to find the most suitable research direction.

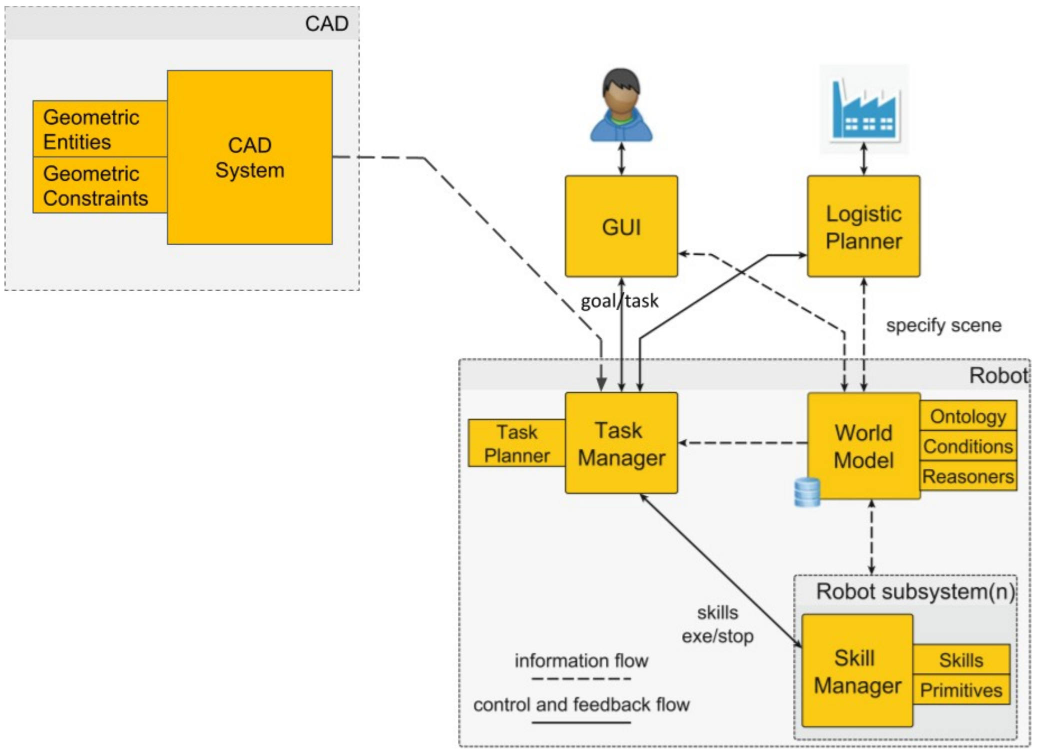


Figure 3.1: Overview of Robotic architecture, Source: Based on Koubaa (2017), p. 128. (modified)

3.1.1 Data types in the robotic world

Robot knowledge can be subdivided into three main domains as follows:

- continuous
- discrete
- semantic

Continuous data are those data that are directly obtained from the sensor systems. The continuous data that is obtained is used to compute and describe the other aspects of the environment and it's called *discrete data*. And *semantic* data is the abstract data that qualitatively describes certain aspects of the environment.[49]

[49] Conf. Koubaa (2017), p. 127-131.

The world model shown in Figure 4.1 stores the semantic data. The world model is the knowledge base where all the relevant data is stored. It supports the skill manager, logic planner and the other sub-systems and any relevant data that needs to be passed on is provided to these systems.

The world model stores all the relevant information in OWL standard.[50] The OWL format can include geometric constraints between multiple objects. It can then perform constraint-based robotic tasks based on the saved geometric constraints[51]. The OWL ontology files are based on the XML syntax and can be easily shared, altered and extracted.

3.1.2 Selection of a suitable CAD system for data extraction

To represent the geometric models of the assembly, two main mathematical models based on a neutral format can be considered; STEP and IGES. STEP has been developed by the International Organization for Standardization as ISO 10303-21, whereas IGES was defined by the United States National Bureau of Standards as NBSIR 80- 1978.

Even though STEP and IGES formats are neutral file systems, they lack the flexibility to associate easily with various other sources of knowledge. Zha and Du used STEP to construct feature-based models, but it required prior knowledge of assembly relations. This information can only be exported using proprietary CAD data formats and gets lost when they are exported as STEP or IGES[52]. Therefore, it was concluded that a proprietary CAD system will be the most suitable for our use case.

Since there was no possibility to use one of the neutral file formats, the next possible alternative would be to use proprietary CAD systems. For the thesis, six different parametric modelling software's were considered. These systems are Autodesk Inventor[53], Solidworks[54], CATiA[55], NX[56], Solidedge[57] and Creo[58]. The basic constraints like Mate/Align, Tangent, Angle Offset and Distance Offset are common among the various systems.

[50] Conf. Garcıa (2010).

[51] Conf. Perzylo et al. (92015), p. 4197-203.

[52] Conf. Zha & Du (2002), p. 1087-110.

[53] Conf. Autodesk, online source [27.09.2020]

[54] Conf. Systèmes, online source [27.09.2020].

[55] Conf. Systèmes, online source [27.09.2020].

[56] Conf. Inc., S. P. L. M. S. Inc., online source [27.09.2020].

[57] Conf. Inc, S. P. L. M. S. Inc., online source [27.09.2020].

[58] Conf. PTC Corporation, online source[27.09.2020]

After analysing all relevant data Autodesk Inventor was selected as the chosen CAD system to build the planning system on. One of the main reasons for choosing Autodesk Inventor was the availability of structured documentation regarding the Application Programming Interface (API) system. Most CAD systems have poor support for the development and there were hardly any resources available in this direction to further develop a planning system. Autodesk Inventor has a better support and knowledge base and hence was chosen as the desired CAD system.

3.1.3 Object models in Inventor

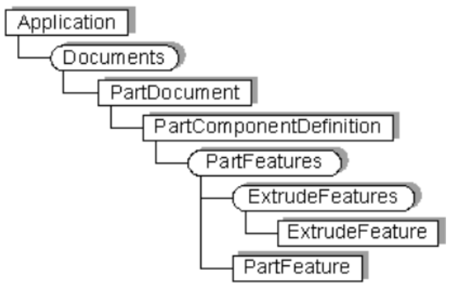


Figure 3.2: Object model in Autodesk Inventor, Source: Autodesk® Inventor® API Object Model reference document, online source [12.09.2020].

The object model is defined as a hierarchical diagram that illustrates the relationships between all the different objects that are defined inside Autodesk Inventor.[59] Objects can be defined as the features that are used inside the various application environment to define and manipulate the design of the drawing, or the part or an assembly. For example, a revolve feature that we use inside the part environment can be defined as a programming object. Therefore, to implement our planning system and to collect the relevant data, we use the object model to extract the relevant information from a given assembly. The following are the relevant information that needs to be extracted to complete the assembly plan.

1. Constraints.	4. Component identity
2. Joints	5. Pose and orientation
3. Component references	6. Bounding box for each component

[59] Conf. Autodesk Inventor, online source [27.09.2020].

3.2 Data representation in CAD systems

The following section describes the models that are used for data representation inside a CAD system. The mathematical model necessary to describe the geometric data is first introduced. The various topological and geometric entities are then defined. Finally, the geometric constraints that relate to each individual part and subassemblies are defined.

3.2.1 Boundary representation

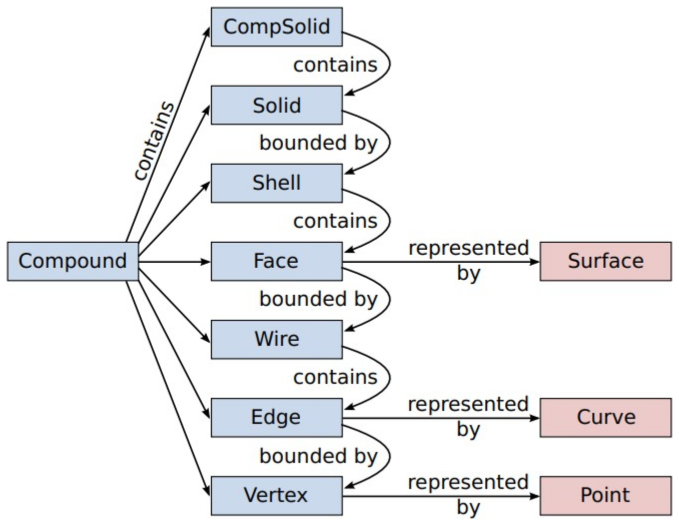


Figure 3.3: Overview of the BREP structure. Topological entities- blue, and geometric entities- red. Source: Based on Perzylo et al. (2015), p. 4197-203

A Boundary Representation (BREP) is a mathematical model that is used to represent primitive geometric components like points, curves and surfaces.[60] Figure 3.3 describes how the data model is represented. The data model is built as *topological entities* which are displayed in blue and *geometric entities* which are displayed in red. The numerical data representing the model is held by the *geometric entities* and the *topological entities* associate them and organise them in a hierarchy. The topological entities and geometric entities are described in detail below.

[60] Conf. Perzylo et al., *supra* note 51, p. 4197-203.

3.2.2 BREP topological entities

The topological entities represented by the BREP standard are classified into eight types:

- vertex
- edge
- wire
- face
- shell
- solid
- compsolid
- compound

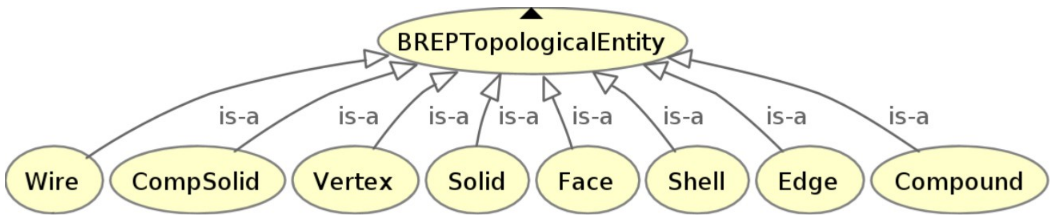


Figure 3.4: Taxonomy of BREP Topological entities, Source: Based on Perzylo et al. (2015), p. 4197-203

A point is used to depict a vertex. Similarly, a curve is depicted using an edge that is defined by two vertices. A set of adjoining edges makes up a wire. A closed wire defined by a surface represents a face and a set of adjacent faces makes up a shell. A solid is defined when the faces of a shell form a closed body. When the solids share adjoining faces, they can be further categorised into compsolids.[61]

3.2.3 BREP geometric entities

The geometric entities are the next level of topological entities. They are made up of three different kinds of entities, which are points that represents a 0-dimensional geometry, curves that represents a 1- dimensional geometry and surfaces that represent a 2- dimensional geometry.

The curves can be further classified into unbounded curves and bounded curves as shown in Figure 3.5. The types of bounded and unbounded curves are given below.

- Bounded curves: Bezier curves, bspline curves, circles, or ellipses.
- Unbounded curves: lines, parabolas or hyperbolas.

[61] Conf. Perzylo et al. (2015), p. 4197-203.

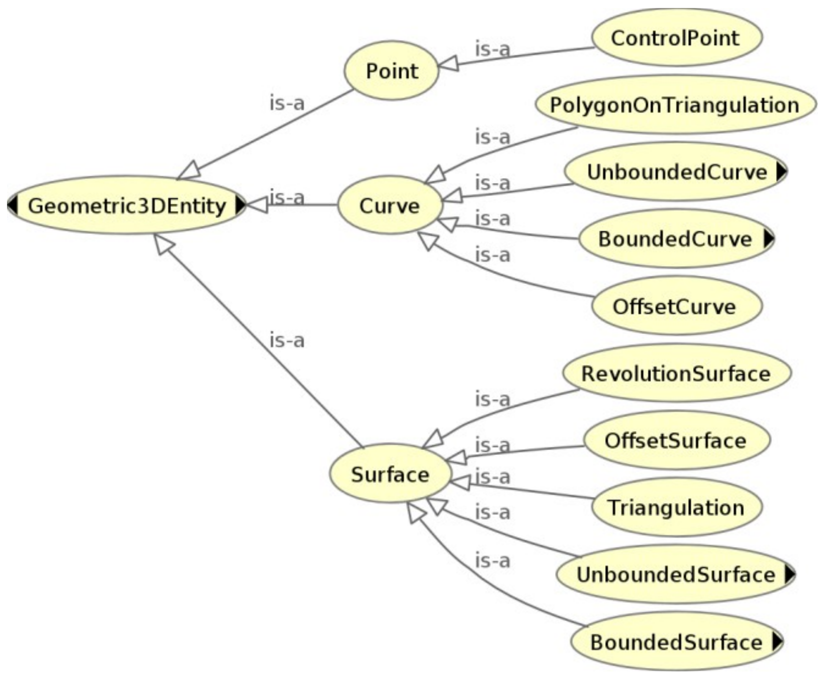


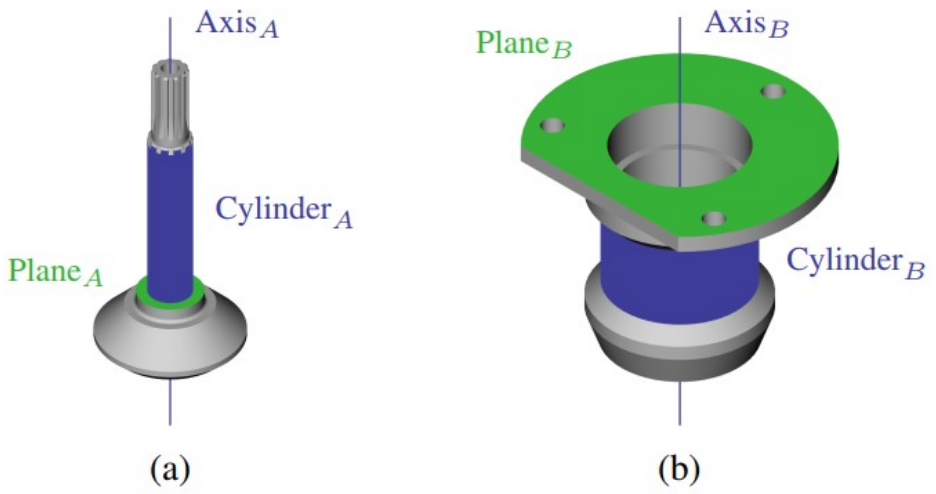
Figure 3.5: Taxonomy of geometric entities, Source: Based on Perzylo et al. (2015), p. 4197-203

When curves are extruded surfaces are formed. Similar to a curve, a surface can be made up of either an unbounded surface or a bounded curve as shown in Figure 3.5. The types of bounded and unbounded surfaces are given below.

- Bounded surfaces: B-spline surfaces, bezier surfaces, spheres, toruses
- Unbounded surfaces: Planes, cones, cylindrical surfaces

As discussed in Section 3.1.2 the neutral file formats are unable to associate additional information as described by the semantic data model. To overcome the challenge, it was decided to use Autodesk Inventor to describe the additional data. The geometric constraints must be first defined to do so. The geometric constraints are component relationships that are defined between different geometric entities like points, curves, and surfaces of the individual parts.

When a geometric constraint is defined, it always connects two or more different geometric entities. There are two different types of geometric entities that need to be defined when a constraint is defined: a parent entity with a defined pose and a constrained or child entity whose pose depends on the parent entity and the constraint itself. Two examples are described below to explain the constraint formulation.



CylinderCylinderConcentricConstraint(Cylinder$_A$, Cylinder$_B$)
PlanePlaneCoincidentConstraint(Plane$_A$, Plane$_B$)

Figure 3.6: Geometric constraints in an assembly task, Source: Based on Perzylo et al. (2015), p. 4197-203

- Cylinder-Cylinder Concentricity Constraint: The precondition for this constraint to exist is the presence of two cylindrical entities. In this case, if we consider part (a) as the parent component and part (b) as the child, the axis of the constrained geometry needs to align with the axis of the base geometry to form a cylinder-cylinder concentricity constraint.
- Plane-Plane Coincidence Constraint: The precondition for this constraint to exist is the presence of two planar entities. In this case, the distance and the rotation of the constrained geometry along the normal vector of the base geometry need to be constrained.[62]

3.2.4 Task specification based on geometric constraints

The relationship constraints defined between the parent and child components can be used to define a robotic task. We can use this data to calculate the assembly pose and orientation. All the necessary geometric constraints need to be described inside the CAD model, to enable smooth manipulation of the objects by the industrial robot.

[62] Conf. Perzylo et al. (2015), p. 4197-203

The proposed conceptual model aims to correctly identify the constrains and provide the task planner with the necessary data contained in the BREP topological description of the CAD model that is connected by the constraints. This data is necessary for the autonomous assembly planning.

3.2.5 Constraint-based robot control

The assembly plan that is created based on the geometric constraints are then combined with the description of the robotic environment and the robotic system including the various manipulation tools and devices.[63] The work cell should also be defined which defines all the related exterior elements like the sensor system, kitting scene, robots, and the workbench layout.

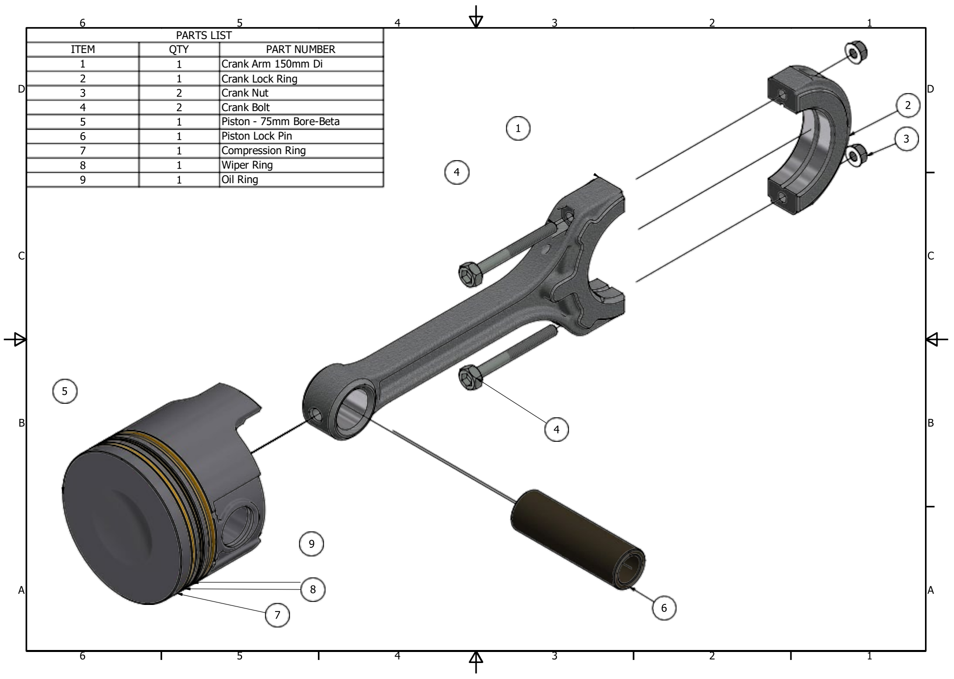
The geometric constraints of the CAD model are analysed to generate the task specification. The robot's poses are then deducted from the task's geometric constraints. Each task instance is thus linked to a particular constraint or a group of constraint which is used to optimize the robots motion or the assembly plan without interfering with the given constraints.

3.3 Assembly sequence planner

The assembly sequence planner is the reasoner that is used to automate the assembly process in the autonomous assembly. To plan for the assembly, the reasoner needs to collect all the data mentioned in the section above to successfully reason between the parts to derive an optimal plan. This section talks about the three main constituting elements of the assembly planning mechanism. They are as given below:

- Assembly relationship matrix
- Assembly sequence planner
- Assembly sequence generation

[63] Conf. Perzylo et al. (2015), p. 4197-203.



PARTS LIST		
ITEM	QTY	PART NUMBER
1	1	Crank Arm 150mm Di
2	1	Crank Lock Ring
3	2	Crank Nut
4	2	Crank Bolt
5	1	Piston - 75mm Bore-Beta
6	1	Piston Lock Pin
7	1	Compression Ring
8	1	Wiper Ring
9	1	Oil Ring

Figure 3.7: Example assembly of an 11 component connecting rod, Source: Own rendering based on Ou & Xu (2013), p. 1053-67. (modified)

3.3.1 Assembly relationship matrix

$$
\begin{array}{c|ccccccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
\hline
1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
2 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
5 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
8 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
9 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
10 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
11 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{array}
$$

Figure 3.8: Assembly relationship matrix, Source: Based on Ou & Xu (2013), p. 1053-67. (modified)

The assembly relationship matrix is a square matrix that is defined by the number of parts of an assembly. The relationships between two different parts are represented by a numerical value. There can be multiple assembly constraint existing between two components.

The value "0" represents that the two parts do not have an existing constraint between them. A value of "1" represents that there is one existing relationship between the two parts. And finally, a value ">1" indicates that there exist multiple constraints between the parts.[64]

The matrix shown above represents an assembly relationship matrix. An example of a piston rod as shown in Figure 3.7 is used to explain the assembly relation matrix. The assembly consists of 11 components as shown in Table 3.1. An 11x11 matrix is formed between each part of the assembly and a value of 1 is populated in all the cells where there exist a relationship between another component. The diagonal elements of the matrix are always going to be zeros. Once the matrix is generated it is easy to identify which component is related to which another component.

Table 3.1: Component list for the piston and connecting rod assembly

Component No.	Component Name
1	Crank Arm 150mm Di
2	Crank Lock Ring
3	Crank Nut A
4	Crank Nut B
5	Crank Bolt A
6	Crank Bolt B
7	Piston- 75mm Bore-Beta
8	Piston Lock Pin
9	Compression Ring
10	Wiper Ring
11	Oil Ring

3.3.2 Assembly sequence planner

To generate an assembly sequence plan, we use a relationship reasoning mechanism as shown in Figure 3.8. The reasoning mechanism determines the order in which parts must be placed on top of each other or in other words the degree at which each part can be assembled between components. For example, if we consider a bolt with a few washers of variable thickness and each of the washer must be placed in a particular order. In this case, the relationship reasoning mechanism identifies each component individually by using its component identity details and they are reasoned on their assemblability using the offset parameter. Each part is placed from the parent object at varying offsets

[64] Conf. Ou & Xu (2013), p. 1053-67.

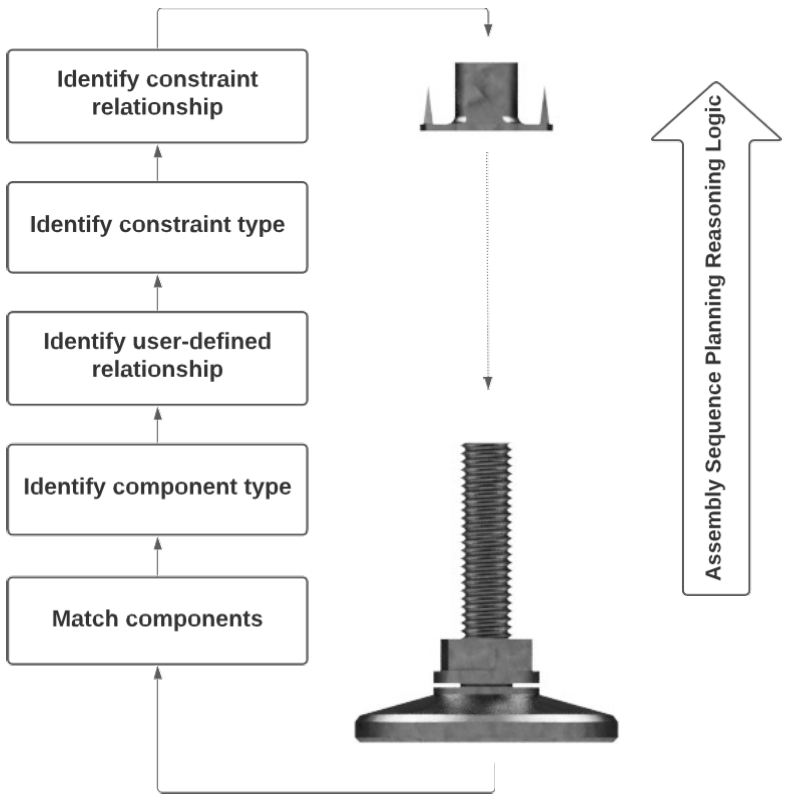

Identify constraint relationship
Identify constraint type
Identify user-defined relationship
Identify component type
Match components
Assembly Sequence Planning Reasoning Logic

Figure 3.8: Assembly sequence planning mechanism, Source: Own rendering based on Ou & Xu (2013), p. 1053-67 (modified).

and this offset is used to identify the order in which every single washer has to be assembled.

The sequence flow of Figure 3.8 is as follows. The components or parts in an assembly is classified into two types, grounded component or a fixed component and a standard component. Classification of the components help in the analysis of the base component for the decision-making mechanism. The grounded component is used as the base of the tree and every other part are built on top of it. The offset act as the user-defined relationship and aids in building the order in which multiple child's has to be placed into a single parent component. The constraint type is then identified which is used to determine the type of mechanism the components form with each other and help determine the motion axis. The constraints between the parent and the child part are used to determine the feasibility and of the assembly sequence.[65]

[65] *Ibid.*, p. 1053-67.

3.4 Assembly graph

The assembly graph is generated as a decision tree as described in Section 2.1.3. The assembly graph consists of nodes that identify every individual component and a direction that represents the child of the parent component. The relation between the two components is identified and represented on the graph. The Figure 3.9 shows an example assembly graph of a clutch plate assembly.

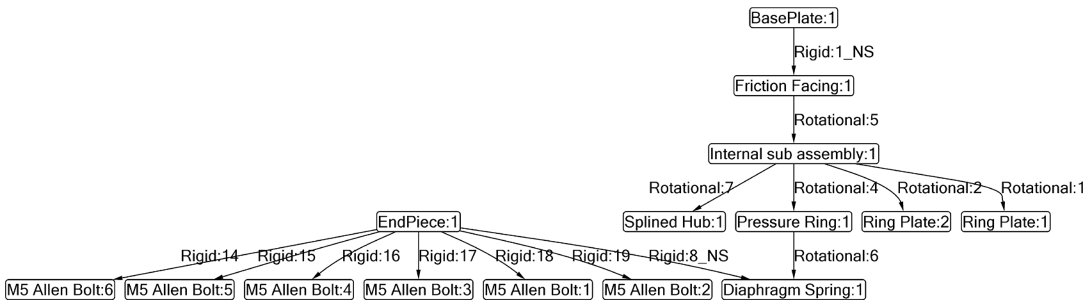


Figure 3.9: Assembly graph based on a clutch plate assembly, Source: Generated using the Inventor assembly exporter.

3.4.1 Assembly sequence generation

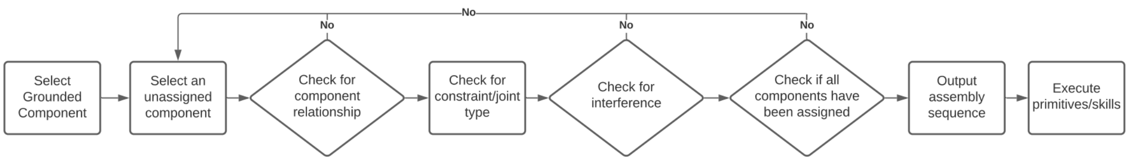

Select Grounded Component
Select an unassigned component
Check for component relationship
Check for constraint/joint type
Check for interference
Check if all components have been assigned
Output assembly sequence
Execute primitives/skills
No
No
No
No

Figure 3.10: Constraint relations assembly sequence planning algorithm, Source: Based on Ou & Xu (2013), p. 1053-67. (modified)

The assembly sequence is generated based on the flowchart as shown in Figure 3.10. The first step in the process is to identify the grounded component. The generation process starts when the first component is selected which is adjacent to the grounded component. The presence of a relationship between the two selected components are checked and then the type of joint or constraint between them is identified. Then the algorithm checks for interference between the two parts and this process is continued until all components have been assigned.

3.5 Data exchange framework

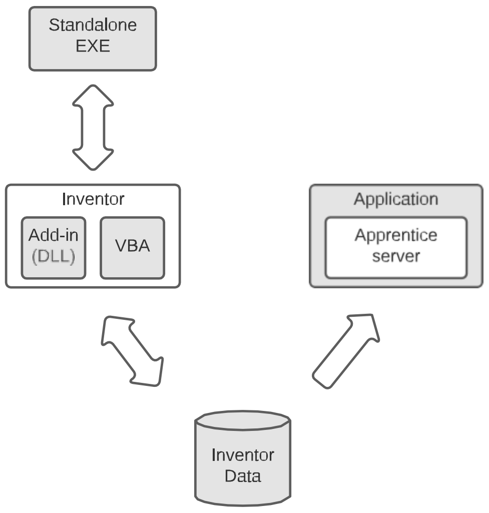


Figure 3.11: Inventor Application Programming Interface, Source: Inventor API User's Manual, online source [27.09.2020].

The proposed system uses the Autodesk Inventor API to provide functions for analysing and collecting the various data that is associated with the defined assembly. The assembly relationship information can be automatically extracted from the Assembly model using the Inventor API.[66]

The framework is developed as a standalone application and connects to Autodesk Inventor. The procedures are developed using C# and the Application Programming Interface (API) is used to manage the information that is present in the assembly model of the CAD system. Even though we have developed this framework for Autodesk Inventor, it can be further developed for any standard CAD system using the same procedure. Figure 3.12 shows the developed standalone application that is used to generate the assembly graph and extract all the relevant data sets for the task planner.

3.6 System implementation

Once the assembly plan is generated as explained in Section 3.4.1 the primitives or the skills are executed. The skills or the primitives are all the necessary

[66] Conf. Autodesk Inventor, *supra* note 59.

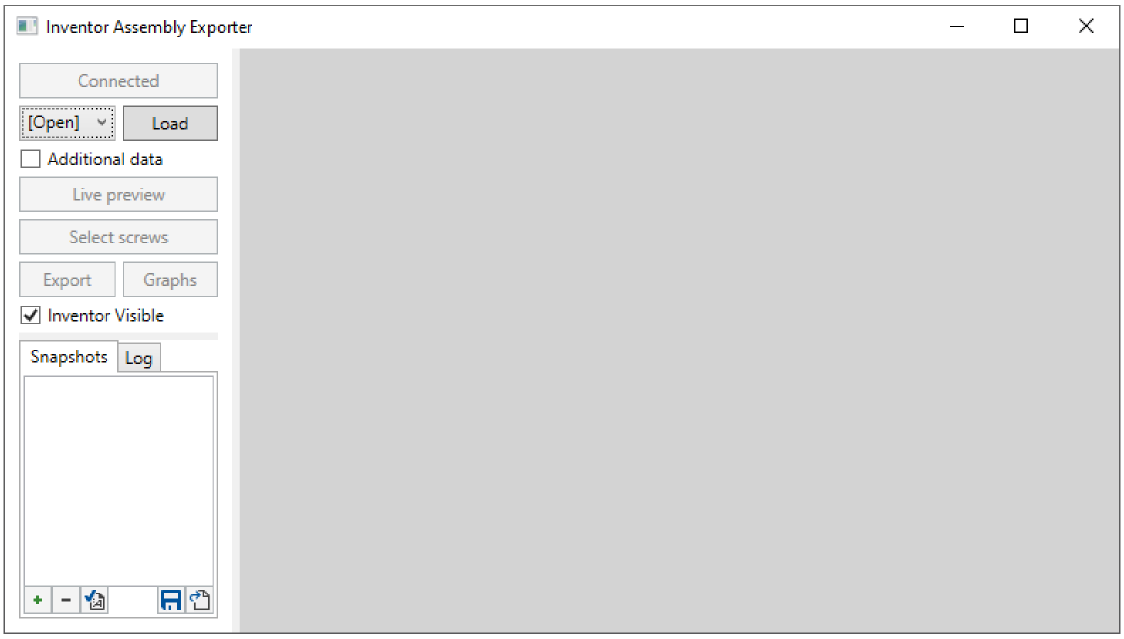


Figure 3.12: Inventor assembly exporter, Source: Based on the developed data exporter.

actions that the manipulators perform to sequentially build up the assembly. The methodology explained in section 3.1 can be expanded to work with any CAD packages available on the market. In our particular use case, we have developed the system to work with Autodesk Inventor. It needs to be pointed out that, in order for the system to work with other CAD packages necessary modifications needs to be made and the functions and the API that is needed for the other CAD software systems will vary depending on the type of application that is being used. The system utilizes the Autodesk Inventor API to provide the necessary functions for collecting and analyzing the assembly components automatically.

In order to validate the developed assembly sequence generation algorithm, we first analyse the WRS Belt drive assembly followed by four use cases as described below

- Use case I: Box world
- Use Case II: Toggle switch
- Use Case III: Perfume bottle
- Use Case IV: Clutch assembly

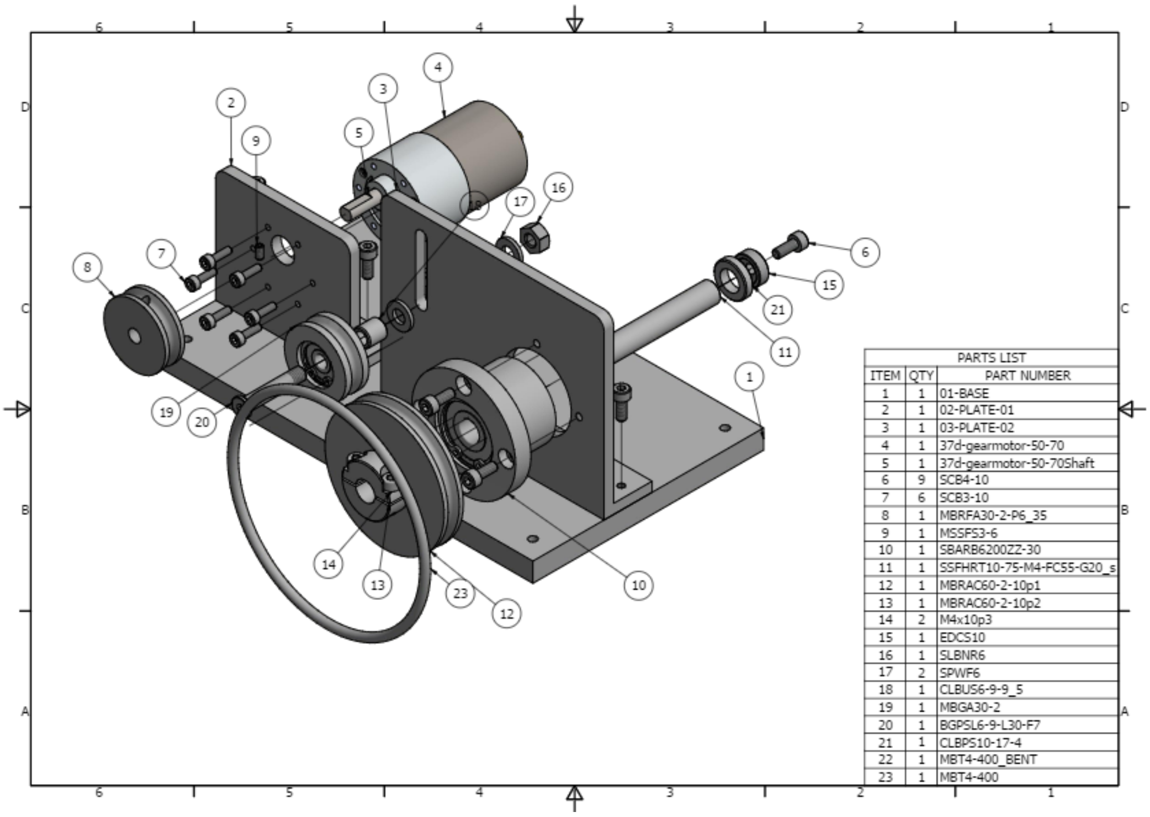
All the assembly models were created in Autodesk Inventor with some of the standard parts being downloaded from the vendor websites. To preserve original modelling features and the design intent, all the files are saved in native inventor file format (.ipt, .iam).

To successfully build the assembly, the following conditions must be followed by the designer who builds the assembly so that the sequence planner does not fail. Without following these rules, the planner is bound to fail and these are the core elements to ensure successful generation of the plan. The conditions have been defined as the “Rules to follow to ensure the successful generation of assembly plan”.

Rules to follow to ensure successful generation of the assembly plan

1. Parent-child order.
2. Always keep in mind that the child is selected first when building the constraints.
3. The Assembly has to be placed to the positive X, Y and Z quadrant.
4. The first piece in the tree must be a grounded object.
5. In terms of relations we only allow Joints of type Rigids and Rotational.
6. Rigids are used to represent the screwing relations inside the assembly.
7. Rigid joints that are used for non-screwing parts should be appended with NS in the end.
8. Parts/Subassemblies that already comes preassembled (or contains a pre satisfied relation) has also to be represented by appending the relation names with a PRESAT (if the relation needs to be changed).
9. The screwing relation always goes between the screw and the part to which the screw is connected (not to the part which is adjacent to or in between the screw and the parent) and this relation is represented using the offset distance between the part and the screw.
10. Bottom-up Assembly approach. (used mainly because we use a lot of pre-assembled or pre-designed standard parts and subassemblies.
11. Every part that is placed in the assembly must be placed taking into consideration how the part is grabbed from the kitting tray. (meaning the direction always pointing in the opposite direction from where the part is placed.
12. GRABPOSE is given by the part file and always coincides with the origin of the part file.
13. The coordinate system in the kitting tray for the parts should also (Z-axis) point opposite to the direction from where the part is grabbed.
14. Origin is always below the kitting tray.

4 Implementation and validation



PARTS LIST		
ITEM	QTY	PART NUMBER
1	1	01-BASE
2	1	02-PLATE-01
3	1	03-PLATE-02
4	1	37d-gearmotor-50-70
5	1	37d-gearmotor-50-70Shaft
6	9	SCB4-10
7	6	SCB3-10
8	1	MBRFA30-2-P6_35
9	1	MSSFS3-6
10	1	SBARB6200ZZ-30
11	1	SSFHRT10-75-M4-FC55-G20_s
12	1	MBRAC60-2-10p1
13	1	MBRAC60-2-10p2
14	2	M4x10p3
15	1	EDCS10
16	1	SLBNR6
17	2	SPWF6
18	1	CLBUS6-9-9_5
19	1	MBGA30-2
20	1	BGPSL6-9-L30-F7
21	1	CLBPS10-17-4
22	1	MBT4-400_BENT
23	1	MBT4-400

Figure 4.1: Assembly diagram with parts list for WRS 2018 belt drive unit, Source: Rendering from own assembly drawing, based on the industrial robotics assembly challenge rulebook (2020), p.44-69

The main goal of this chapter is to verify the methodology described in the previous chapter and to arrive at a conclusion if the proposed framework is ideal. The belt drive unit is used as an example to test the method. The belt drive unit along with the flexible assembly provides enough corner cases to include a broad spectrum of general assembly use cases in the industry.

The main features of this sub-assembly are listed below.

- The subassemblies consist of around 20 parts.

- The presence of flexible parts.
- A major number of parts in the belt drive unit have high tolerances.
- Standard parts like set screw, nuts and washers.
- Simultaneous manipulation of multiple robotic arms.
- Vertical and horizontal assembly.
- When M6 nut and M6 bolt needs to be assembled; one part should be held by the robotic arm and the other screwed simultaneously. The robotic arm also has to approach horizontally in this case.
- The setscrews are pre screwed into the pulley and connects the pulley to the motor shaft.
- The shaft is fastened by using M4 bolt (Part #13) and the end cap (Part #15). For the subassembly to be successfully completed the end cap is placed on the shaft end and fastened with the M4 bolt.

4.1 Problem definition

The problem aims at autonomously assembling the belt drive unit which consists of 23 parts along with a final flexible part which is an elastic rubber band.

The assembly task is the most challenging task out of all the different tasks inside the Industrial Robotics Category. It requires multiple part reorientations and there are multiple insertion directions which make the assembly quite complex and challenging.

Table 4.1: Assembly operations or skills required to finish the assembly process

hole-on-peg	screwing
peg-in-hole	inserting
tightening	belt hooking
fit-on-shaft	screwing into a tapped hole
ring through shaft	fastening of a nut into a bolt

The following different assembly operations as shown in Table 4.1 are identified to complete the belt drive assembly.

4.2 Requirements list

This section lists the set of requirements that is to be satisfied to fulfil complete working of the developed model.

- All the individual components in the assembly should be first identified.
- The number of assembly relations between different components should be identified.
- Based on the identified relationship and part offsets the assembly graph should be generated.
- The generated assembly graph is used to derive the assembly plan.
- The assembly plan is passed on to the skill manager which acts on the plan and produces the required final assembly.

Table 4.2: Parts list for WRS2020 Assembly Task, Source: Based on the industrial robotics assembly challenge rulebook (2020), p.69

Assembly Task			
	Parts list for WRS2020 Assembly Task	2020.Jan.14	
No.	Name of part	Note.	Qty.
1	01-BASE	Base plate	1
2	02-PLATE	Output shaft fixing plate	1
3	03-PLATE2	Motor fixing plate	1
4	04_37D-GEARMOTOR-50-70	70:1 Metal Gearmotor 37Dx54L mm 12V (Helical Pinion)	1
5	05_MBRFA30-2-P6	Pulley for Round Belt (4mm) - Setscrew P.D. 30mm	1
6	06_MBT4-400	Polyurethane round belt (welded joint product) P.D. 4mm L=400mm	1
7	07_SBARB6200ZZ_30	Bearings with Housings (Double Bearings)	1
8	08_SSFHRT10-75-M4-FC55-G20	Drive shaft (Straight) D10h7	1
9	09_EDCS10	End Cap for Shaft	1
10	10_CLBPS10_17_4	Bearing Spacers For Inner Ring (output pulley)	1
11	11_MBRAC60-2-10	Pulley for Round Belts Clamping Type P.D. 60mm	1
12	12_CLBUS6-9-9.5	Bearing Spacers For Inner Ring (tension pulley)	1
13	13_MBGA30-2	Idler for Round Belt - Wide	1
14	14_BGPSL6-9-L30-F7	Bearing Shaft Screw	1
15	15_SLBNR6	M6 Hex Nut (Fixing for idler shaft)	1
16	16_SPWF6	M6 Flat Washer (Fixing for idler shaft)	2
17	17_SCB4-10	10mm M4 Socket Head Cap Screw (metric coarse thread)	9
18	18_SCB3-10	10mm M3 Socket Head Cap Screw (metric coarse thread)	6
19	19_MSSFS3-6	6mm M3 Hex Socket Set Screw (metric coarse thread)	1
20	20_TW1004	Clutch lock terminal block (compact)	1
21	21_H0.5/14D	Weidmuller Wire-end ferrule	2
22	22_NAUL1015_22_BK	NAUL1015 UL compliant wire (black)	1
23	23_NAUL1015_22_R	NAUL1015 UL compliant wire (red)	1

4.3 Assembly components

The Assembly challenge at the WRS 2018 consists of 23 different components or parts that are used in varying numbers to build up the assembly.

There are three non-standard components inside the assembly constituting the base plate and two other mounting plates. Apart from these three parts, all the other parts are standard parts that are mass manufactured. All the standard parts are available online at misumi all around the world.[67] The standard parts include common assembly components like bolts, screws/nuts, spacers, pulleys, end caps and other parts like belt drive unit, pulleys, flexible cables, bearings with housings and motor.

4.4 Subassembly components

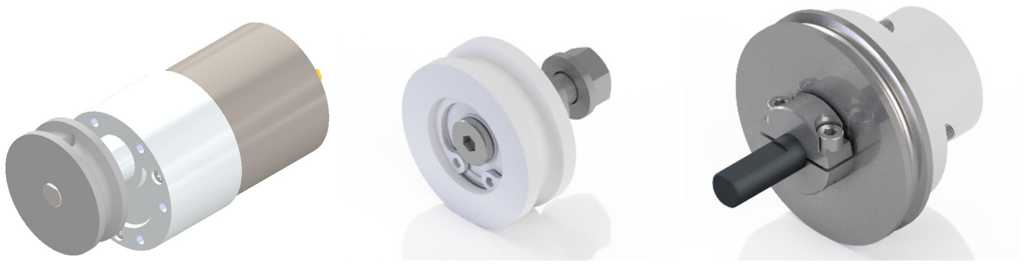


Figure 4.2: The three subassembly components in the assembly task, Source: Rendering from own assembly drawing, based on the industrial robotics assembly challenge rulebook (2020), p.44-69

There are three main sub-assemblies that are assembled on the main assembly. It includes tasks that contain assembly subtasks and disassembly subtasks. The three main subassemblies are shown in Figure 4.2. It consists of a gearmotor subassembly as shown in Figure 4.2 (a), an idler subassembly for the flexible belt as shown in 4.2 (b), and the main pulley subassembly as shown in 4.2 (c).

4.5 Kitting scene

The part orientations and grab positions have been simplified by the use of a kitting tray as shown in Figure 4.3. A few of the parts had additional supports and fixtures that had been added in place to ensure that the manipulator grabs the parts in the right orientation.

[67] Conf. Misumi, online source [27.09.2020]

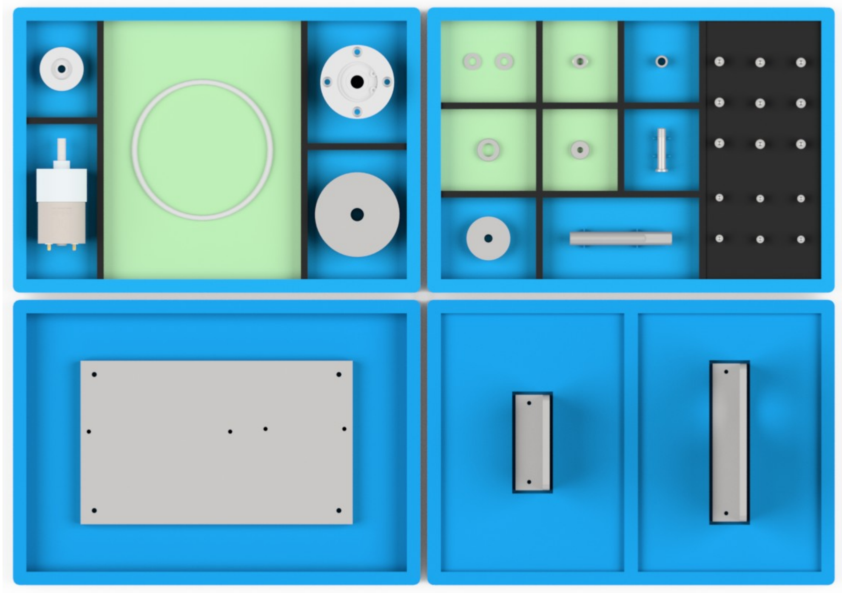


Figure 4.3: Kitting tray with components for the belt drive unit, Source: Rendering from own assembly drawing, based on the industrial robotics assembly challenge rulebook (2020), p.12-14

The initial picking task is neglected in this assembly and the process starts directly at the kitting tray for simplification. This is done to avoid bin picking and the complications involved with the picking of the different parts.

4.6 Data set

The data set consists of the necessary data points that have to be extracted from the CAD assembly model. It consists of data's like the component identity details that help identify each individual component to various assembly relationships, poses and orientation, and component initial positions. The relevant data necessary for the task planner are described in detail below.

Component identity details

The component identity details are the placeholder for determining what kind of a part is used and is usually identified with a unique ID. The component ID is unique and is used to differentiate between similar parts that are used throughout the assembly.

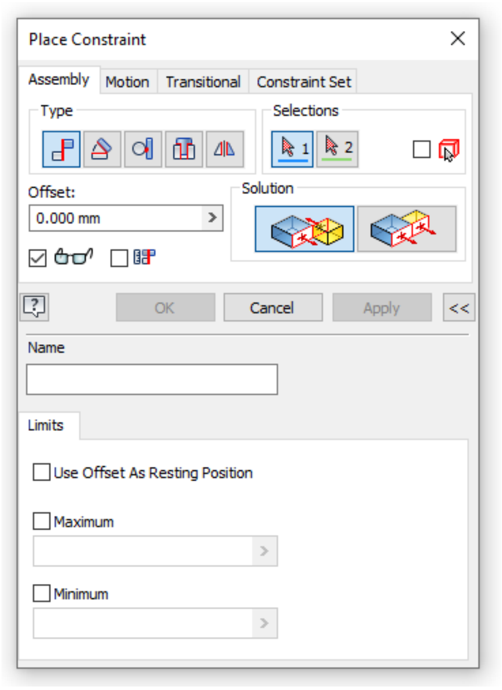


Figure 4.4: Assembly constraints in Autodesk Inventor, Source: Autodesk® Inventor

Constraints

Assembly constraints are used to limit the degrees of freedom of a component in the assembly environment.[68] There are two different kinds of relationships when it comes to constraints in Autodesk Inventor. The basic type of constraint used defines how the child component interact with its parent component along with the direction in which both the components lie with respect to each other. The basic constraint elements constitute constraint types mate, angle, tangent, insert and symmetry.

Types of joints

The second type of constraints are mechanism constraints that defines the kinematic relationship between the components. The number of Degree of Freedom (DoF) the component has is defined along with the possible directions in which the part can move. When all the degrees of freedom are arrested, we end up with a rigid joint. The rigid functions pass on the information to the assembly that this part has no mechanical functions. All the rest of the constraints can have their degrees of freedom varying from 1 to 6.

[68] Conf. Autodesk, *supra* note 53.

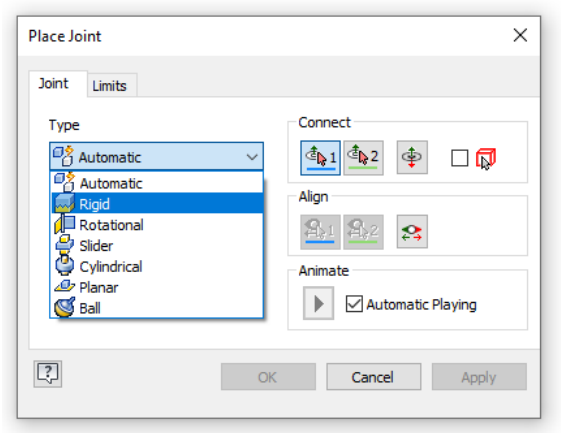


Figure 4.5: Assembly joints in Autodesk Inventor, Source: Autodesk® Inventor

The kinematic constraints in Autodesk Inventor constitute rigid, rotational, slider, cylindrical, planar and ball joints.[69]

Component references

When an assembly relationship is defined between two parts, the child and parent parts each reference component geometries to create the required constraint. The type of constraint along with the component references are used to determine the assembly direction. For example, an assembly consisting of a peg and hole represented by parts A and B, where A is inserted into B, the references used are the outer diameter of A and the inner diameter of B. The type of fit between the two parts can be easily determined by analysing the component geometries or features. Other useful information can also be derived like offset date to determine if there is direct contact between two components or interferences.

Component initial positions

To determine the spatial constraints the initial position of the component with respect to the coordinate system is used. The initial position is determined by the designer when he places the component inside the assembly. There are two frames of references when it comes to the position data. A global position is defined for each individual component with respect to the assembly as a whole. There is also a relative position of components that are defined with respect to one another.

[69] *Ibid*.

Pose and orientation

Pose and orientation are the two main elements that represent an object in 3d space inside a CAD workspace. A coordinate system is used to define the position and orientation of an object in space. In the case of Inventor workspace the position of the coordinate system is represented by (x, y, z). The orientation of the object on the other hand is defined by specifying the direction of the x, y and z axes. The inventor framework uses a rectangular coordinate system.

Vector objects

A vector is the most convenient way to specify a magnitude and direction in a CAD system. It mainly consists of three values: x, y and z components. The three values here represent the three coordinates and is usually given by a matrix (x, y, z). A 2d vector is represented by x and y components. The most common use of a vector is to define the movement of an object.

Whenever a vector is used to express the direction alone, then we use a Unit vector whose magnitude is always 1. In our implementation, we use the unit vector in several cases in order to identify the orientation information of the components. For example, imagine a cone whose axis vector returns a unit vector object. This vector defines the direction of the axis of the cylinder.

Bounding box from an object

The bounding box of a part or a component is represented by the three values x, y and z. It is the overall length(x), width(y) and height(z) value of a part or an assembly file. The bounding box value is relevant in our case because it is used to calculate the collisions between individual parts and the point at which the object has to be grabbed.

Parent and child relationship

The assembly environment is represented using a parent-child relationship. This parent-child relationship is used to build the relationship node diagram and it always follows a tree structure. In order to generate a successful assembly plan, the assembly should be built by following the rules specified in Section 3.6.

Assembly snapshots

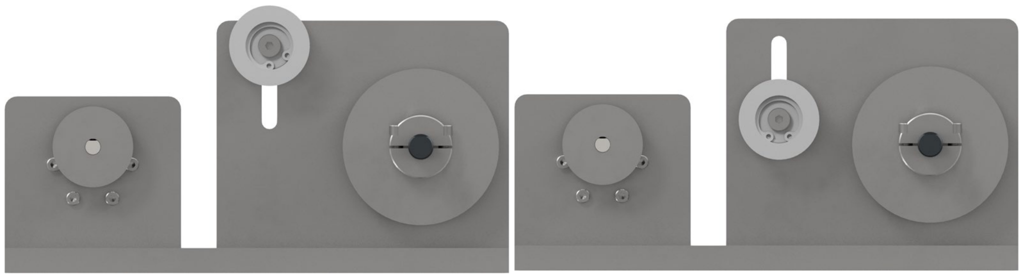


Figure 4.6: Belt drive unit assembly snapshots (a) Configuration 1 (b) Configuration 2, Source: Rendering from own assembly drawing, based on the industrial robotics assembly challenge rulebook (2020), p.44-69

As discussed in the introduction of this section, this task requires assembly and disassembly to successfully complete the whole assembly process. We achieved this by using snapshots, where two product configurations are saved, and two different assembly plans are generated for the different instances. Later they are executed to fully assemble the belt drive unit. The Figure 4.6 shows the two snapshots that were required to move the sub-assembly from the top position to the bottom position.

Flexible assembly

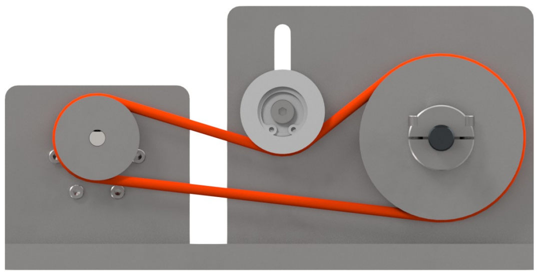


Figure 4.7: The belt drive unit used for the assembly challenge with the flexible part, Source: Rendering from own assembly drawing, based on the industrial robotics assembly challenge rulebook (2020), p.44-69

The most challenging part of the assembly is to put together the rubber ring which is the last step of the assembly. Since it required a special gripper and additional setup, it was decided to add it to the future expansion of the project. The important thing to note in the case of the flexible rubber band is that the subassembly that is located in the middle of the belt drive unit has to push against it after the rubber band is assembled onto the motor and the rotating pulley. The subassembly in the middle has to then press against it a force while still screwing the nut and holding the bolt on which the whole subassembly is stacked.

4.7 Assembly graph

All the data set describes above are then used to generate the assembly graph of the required assembly model. All the assembly relations and parent-child order are specified in the graph. The belt drive unit has two assembly configurations before the final assembly state can be achieved. The assembly graph generated using the developed assembly exporter are shown below.

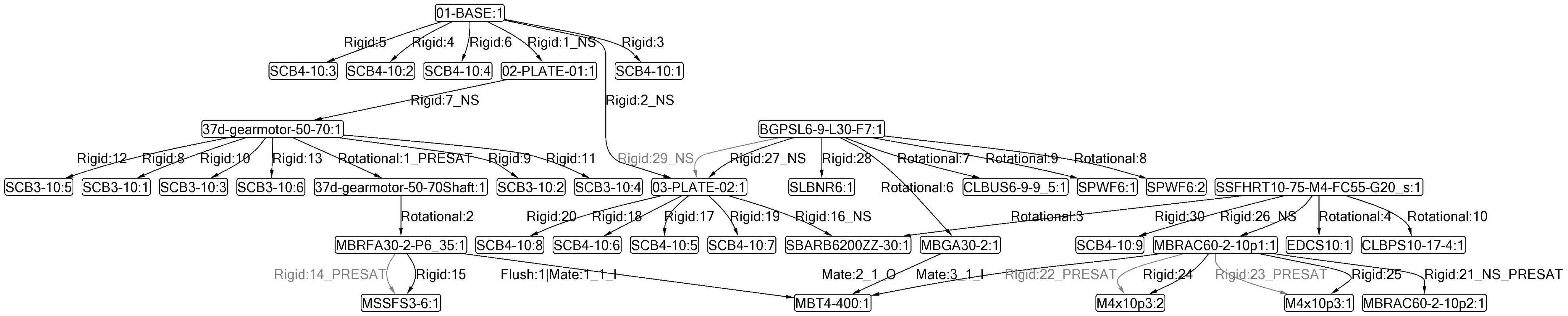


Figure 4.8: Assembly node diagram for the belt drive unit: Configuration 1, Source: Generated using the Inventor Assembly exporter.

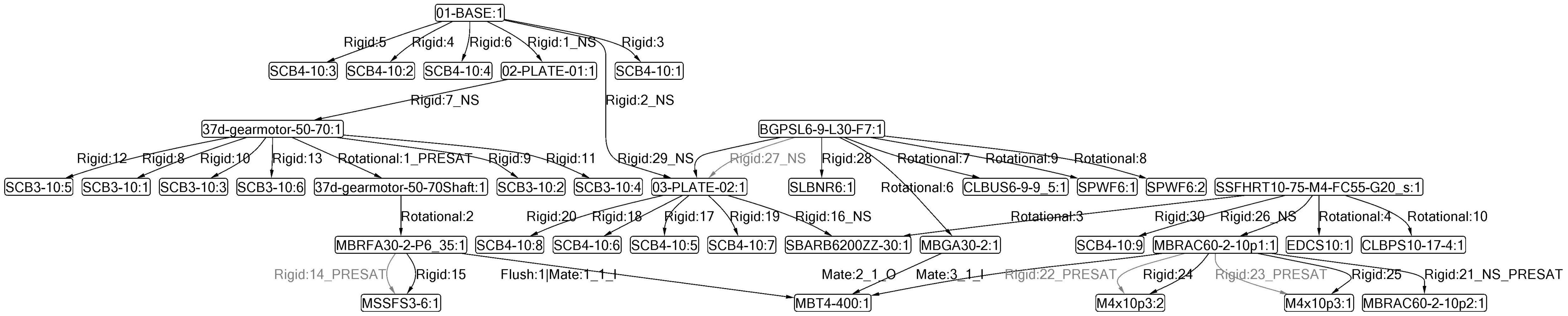


Figure 4.9: Assembly node diagram for the belt drive unit: Configuration 2, Source: Generated using the Inventor Assembly exporter.

4.8 Assembly sequence plan

The assembly graph generated above is used to obtain the optimal assembly sequence plan. The assembly plan for the belt drive unit is shown below.

Assembly sequence plan of belt drive unit

```
    SKILL SKIP      00: Grab Step001_01-BASE+1 and insert into ground
    SKILL EXECUTION 01: Grab Step001_02-PLATE-01+1 and insert into Step001_01-BASE+1 using left_ur5_arm
                        -> Grab Step001_02-PLATE-01+1 using left_ur5_arm
                        -> Insert Step001_02-PLATE-01+1 into Step001_01-BASE+1 using left_ur5_arm
    SKILL EXECUTION 02: Grab Step001_SCB4-10+3 and screw into Step001_01-BASE+1 using right_ur5_arm
                        -> Grab Step001_SCB4-10+3 using right_ur5_arm
                        -> Screw Step001_SCB4-10+3 to Step001_01-BASE+1 using right_ur5_arm
    SKILL EXECUTION 03: Grab Step001_SCB4-10+4 and screw into Step001_01-BASE+1 using right_ur5_arm
                        -> Grab Step001_SCB4-10+4 using right_ur5_arm
                        -> Screw Step001_SCB4-10+4 to Step001_01-BASE+1 using right_ur5_arm
                        -> Let go off Step001_02-PLATE-01+1 with left_ur5_arm
    SKILL EXECUTION 04: Grab Step001_03-PLATE-02+1 and insert into Step001_01-BASE+1 using left_ur5_arm
                        -> Grab Step001_03-PLATE-02+1 using left_ur5_arm
                        -> Insert Step001_03-PLATE-02+1 into Step001_01-BASE+1 using left_ur5_arm
    SKILL EXECUTION 05: Grab Step001_SCB4-10+1 and screw into Step001_01-BASE+1 using right_ur5_arm
                        -> Grab Step001_SCB4-10+1 using right_ur5_arm
                        -> Screw Step001_SCB4-10+1 to Step001_01-BASE+1 using right_ur5_arm
    SKILL EXECUTION 06: Grab Step001_SCB4-10+2 and screw into Step001_01-BASE+1 using right_ur5_arm
                        -> Grab Step001_SCB4-10+2 using right_ur5_arm
                        -> Screw Step001_SCB4-10+2 to Step001_01-BASE+1 using right_ur5_arm
    *LET GO     -> Left Arm Step001_03-PLATE-02+1
    SKILL EXECUTION 07: Grab Step001_37d-gearmotor-50-70+1 and insert into Step001_02-PLATE-01+1 using right_ur5_arm
                        -> Grab Step001_37d-gearmotor-50-70+1 using right_ur5_arm
                        -> Insert Step001_37d-gearmotor-50-70+1 into Step001_02-PLATE-01+1 using right_ur5_arm
    SKILL SKIP      08: Grab Step001_37d-gearmotor-50-70Shaft+1 and insert into Step001_37d-gearmotor-50-70+1
    SKILL EXECUTION 09: Grab Step001_SCB3-10+2 and screw into Step001_37d-gearmotor-50-70+1 using left_ur5_arm
                        -> Grab Step001_SCB3-10+2 using left_ur5_arm
                        -> Screw Step001_SCB3-10+2 to Step001_37d-gearmotor-50-70+1 using left_ur5_arm
    SKILL EXECUTION 10: Grab Step001_SCB3-10+4 and screw into Step001_37d-gearmotor-50-70+1 using left_ur5_arm
                        -> Grab Step001_SCB3-10+4 using left_ur5_arm
                        -> Screw Step001_SCB3-10+4 to Step001_37d-gearmotor-50-70+1 using left_ur5_arm
    SKILL EXECUTION 11: Grab Step001_SCB3-10+6 and screw into Step001_37d-gearmotor-50-70+1 using left_ur5_arm
```

```
                        -> Grab Step001_SCB3-10+6 using left_ur5_arm
                        -> Screw Step001_SCB3-10+6 to Step001_37d-gearmotor-50-70+1
 using left_ur5_arm


  SKILL EXECUTION 12: Grab Step001_SCB3-10+1 and screw into Step001_37d-
gearmotor-50-70+1 using left_ur5_arm
                        -> Grab Step001_SCB3-10+1 using left_ur5_arm
                        -> Screw Step001_SCB3-10+1 to Step001_37d-gearmotor-50-70+1
 using left_ur5_arm
  SKILL EXECUTION 13: Grab Step001_SCB3-10+5 and screw into Step001_37d-
gearmotor-50-70+1 using left_ur5_arm
                        -> Grab Step001_SCB3-10+5 using left_ur5_arm
                        -> Screw Step001_SCB3-10+5 to Step001_37d-gearmotor-50-70+1
 using left_ur5_arm
  SKILL EXECUTION 14: Grab Step001_SCB3-10+3 and screw into Step001_37d-
gearmotor-50-70+1 using left_ur5_arm
                        -> Grab Step001_SCB3-10+3 using left_ur5_arm
                        -> Screw Step001_SCB3-10+3 to Step001_37d-gearmotor-50-70+1
 using left_ur5_arm
                        -> Let go off Step001_37d-gearmotor-50-70+1 with
right_ur5_arm
  SKILL EXECUTION 15: Grab Step001_MBRFA30-2-P6_35+1 and insert into
Step001_37d-gearmotor-50-70Shaft+1 using left_ur5_arm
                        -> Grab Step001_MBRFA30-2-P6_35+1 using left_ur5_arm
                        -> Insert Step001_MBRFA30-2-P6_35+1 into Step001_37d-
gearmotor-50-70Shaft+1 using left_ur5_arm
  *LET GO     -> Left Arm  Step001_MBRFA30-2-P6_35+1
  SKILL SKIP      16: Grab Step001_MSSFS3-6+1 and screw into Step001_MBRFA30
-2 - P6_35 +1
  SKILL EXECUTION 17: Grab Step001_SBARB6200ZZ-30+1 and insert into Step001_03
-PLATE-02+1 using right_ur5_arm
                        -> Grab Step001_SBARB6200ZZ-30+1 using right_ur5_arm
                        -> Insert Step001_SBARB6200ZZ-30+1 into Step001_03-PLATE
-02+1 using right_ur5_arm
  SKILL EXECUTION 18: Grab Step001_SCB4-10+6 and screw into Step001_03-PLATE
-02+1 using left_ur5_arm
                        -> Grab Step001_SCB4-10+6 using left_ur5_arm
                        -> Screw Step001_SCB4-10+6 to Step001_03-PLATE-02+1 using
left_ur5_arm
  SKILL EXECUTION 19: Grab Step001_SCB4-10+8 and screw into Step001_03-PLATE
-02+1 using left_ur5_arm
                        -> Grab Step001_SCB4-10+8 using left_ur5_arm
                        -> Screw Step001_SCB4-10+8 to Step001_03-PLATE-02+1 using
left_ur5_arm
  SKILL EXECUTION 20: Grab Step001_SCB4-10+5 and screw into Step001_03-PLATE
-02+1 using left_ur5_arm
                        -> Grab Step001_SCB4-10+5 using left_ur5_arm
                        -> Screw Step001_SCB4-10+5 to Step001_03-PLATE-02+1 using
left_ur5_arm
  SKILL EXECUTION 21: Grab Step001_SCB4-10+7 and screw into Step001_03-PLATE
-02+1 using left_ur5_arm
                        -> Grab Step001_SCB4-10+7 using left_ur5_arm
                        -> Screw Step001_SCB4-10+7 to Step001_03-PLATE-02+1 using
left_ur5_arm
                        -> Let go off Step001_SBARB6200ZZ-30+1 with right_ur5_arm
  SKILL EXECUTION 22: Grab Step001_MBRAC60-2-10p1+1 and insert into
Step001_SSFHRT10-75-M4-FC55-G20_s+1 using left_ur5_arm
                        -> Grab Step001_MBRAC60-2-10p1+1 using left_ur5_arm
                        -> Grab Step001_SSFHRT10-75-M4-FC55-G20_s+1 using
right_ur5_arm
                        -> Insert Step001_MBRAC60-2-10p1+1 into Step001_SSFHRT10
-75-M4-FC55-G20_s+1 using left_ur5_arm
  SKILL EXECUTION 23: Grab Step001_CLBPS10-17-4+1 and insert into
```

```
Step001_SSFHRT10-75-M4-FC55-G20_s+1 using left_ur5_arm
                    -> Grab Step001_CLBPS10-17-4+1 using left_ur5_arm
                    -> Insert Step001_CLBPS10-17-4+1 into Step001_SSFHRT10-75-
M4-FC55-G20_s+1 using left_ur5_arm
  *LET GO     -> Left Arm Step001_CLBPS10-17-4+1
  SKILL EXECUTION 24: Grab Step001_SSFHRT10-75-M4-FC55-G20_s+1 and insert into
 Step001_SBARB6200ZZ-30+1 using right_ur5_arm
                    -> Insert Step001_SSFHRT10-75-M4-FC55-G20_s+1 into
Step001_SBARB6200ZZ-30+1 using right_ur5_arm
  SKILL EXECUTION 25: Grab Step001_EDCS10+1 and insert into Step001_SSFHRT10
-75-M4-FC55-G20_s+1 using left_ur5_arm
                    -> Grab Step001_EDCS10+1 using left_ur5_arm
                    -> Insert Step001_EDCS10+1 into Step001_SSFHRT10-75-M4-FC55
-G20_s+1 using left_ur5_arm
  SKILL EXECUTION 26: Grab Step001_SCB4-10+9 and screw into Step001_SSFHRT10
-75-M4-FC55-G20_s+1 using left_ur5_arm
                    -> Grab Step001_SCB4-10+9 using left_ur5_arm
                    -> Screw Step001_SCB4-10+9 to Step001_SSFHRT10-75-M4-FC55-
G20_s+1 using left_ur5_arm
  SKILL SKIP      27: Grab Step001_MBRAC60-2-10p2+1 and insert into
Step001_MBRAC60-2-10p1+1
  SKILL SKIP      28: Grab Step001_M4x10p3+2 and screw into Step001_MBRAC60
-2-10p1+1
  SKILL SKIP      29: Grab Step001_M4x10p3+1 and screw into Step001_MBRAC60
-2-10p1+1
                    -> Let go off Step001_SSFHRT10-75-M4-FC55-G20_s+1 with
right_ur5_arm
  SKILL EXECUTION 30: Grab Step001_MBGA30-2+1 and insert into Step001_BGPSL6
-9-L30-F7+1 using left_ur5_arm
                    -> Grab Step001_MBGA30-2+1 using left_ur5_arm
                    -> Grab Step001_BGPSL6-9-L30-F7+1 using right_ur5_arm
                    -> Insert Step001_MBGA30-2+1 into Step001_BGPSL6-9-L30-F7+1
 using left_ur5_arm
  SKILL EXECUTION 31: Grab Step001_CLBUS6-9-9_5+1 and insert into
Step001_BGPSL6-9-L30-F7+1 using left_ur5_arm
                    -> Grab Step001_CLBUS6-9-9_5+1 using left_ur5_arm
                    -> Insert Step001_CLBUS6-9-9_5+1 into Step001_BGPSL6-9-L30-
F7+1 using left_ur5_arm
  SKILL EXECUTION 32: Grab Step001_SPWF6+2 and insert into Step001_BGPSL6-9-
L30-F7+1 using left_ur5_arm
                    -> Grab Step001_SPWF6+2 using left_ur5_arm
                    -> Insert Step001_SPWF6+2 into Step001_BGPSL6-9-L30-F7+1
using left_ur5_arm
  *LET GO     -> Left Arm  Step001_SPWF6+2
  SKILL EXECUTION 33: Grab Step001_BGPSL6-9-L30-F7+1 and insert into
 Step001_03-PLATE-02+1 using right_ur5_arm
                    -> Insert Step001_BGPSL6-9-L30-F7+1 into Step001_03-PLATE
-02+1 using right_ur5_arm
  SKILL EXECUTION 34: Grab Step001_SPWF6+1 and insert into Step001_BGPSL6-9-
L30-F7+1 using left_ur5_arm
                    -> Grab Step001_SPWF6+1 using left_ur5_arm
                    -> Insert Step001_SPWF6+1 into Step001_BGPSL6-9-L30-F7+1
using left_ur5_arm
  SKILL EXECUTION 35: Grab Step001_SLBNR6+1 and screw into Step001_BGPSL6-9-
L30-F7+1 using left_ur5_arm
                    -> Grab Step001_SLBNR6+1 using left_ur5_arm
                    -> Screw Step001_SLBNR6+1 to Step001_BGPSL6-9-L30-F7+1
using left_ur5_arm
                    -> Let go off Step001_BGPSL6-9-L30-F7+1 with right_ur5_arm
  SKILL SKIP      36: Grab Step001_MSSFS3-6+1 and unscrew from Step001_MBRFA30
-2-P6_35+1
  SKILL EDIT      37: Grab Step002_MSSFS3-6+1 and re-screw into
```

```
    Step002_MBRFA30-2-P6_35+1 using left_ur5_arm
                        -> Grab Step002_MSSFS3-6+1 using left_ur5_arm
                        -> Screw Step002_MSSFS3-6+1 to Step002_MBRFA30-2-P6_35+1
    using left_ur5_arm
      SKILL SKIP       38: Grab Step001_M4x10p3+2 and unscrew from Step001_MBRAC60
    -2-10p1+1
      SKILL EDIT       39: Grab Step002_M4x10p3+2 and re-screw into Step002_MBRAC60
    -2-10p1+1 using left_ur5_arm
                        -> Grab Step002_M4x10p3+2 using left_ur5_arm
                        -> Screw Step002_M4x10p3+2 to Step002_MBRAC60-2-10p1+1
    using left_ur5_arm
      SKILL SKIP       40: Grab Step001_M4x10p3+1 and unscrew from Step001_MBRAC60
    -2-10p1+1
      SKILL EDIT       41: Grab Step002_M4x10p3+1 and re-screw into Step002_MBRAC60
    -2-10p1+1 using left_ur5_arm
                        -> Grab Step002_M4x10p3+1 using left_ur5_arm
                        -> Screw Step002_M4x10p3+1 to Step002_MBRAC60-2-10p1+1
    using left_ur5_arm
      UNSUPPORTED      42: Grab Step003_MBT4-400+1 and insert into Step003_MBGA30
    -2+1
                       42: related connection Step003_MBRAC60-2-10p1+1 and
    Step003_MBT4-400+1
                       42: related connection Step003_MBRFA30-2-P6_35+1 and
    Step003_MBT4-400+1
      SKILL UNDO 1ROT 43: Grab Step003_SLBNR6+1 and unscrew from Step003_BGPSL6-9-
    L30-F7+1 using left_ur5_arm
                        -> Grab Step003_SLBNR6+1 using left_ur5_arm
                        -> Grab Step003_BGPSL6-9-L30-F7+1 using right_ur5_arm
                        -> Unscrew Step003_SLBNR6+1 using left_ur5_arm
      *LET GO      -> Left Arm  Step003_SLBNR6+1
      SKILL SKIP       44: Grab Step003_BGPSL6-9-L30-F7+1 and extract from
    Step003_03-PLATE-02+1
      SKILL EDIT       45: Grab Step004_BGPSL6-9-L30-F7+1 and re-insert into
    Step004_03-PLATE-02+1 using right_ur5_arm
                        -> Shift Step004_BGPSL6-9-L30-F7+1 within Step004_03-PLATE
    -02+1 using right_ur5_arm
      SKILL EXECUTION 46: Grab Step004_SLBNR6+1 and screw into Step004_BGPSL6-9-
    L30-F7+1 using left_ur5_arm
                        -> Screw Step004_SLBNR6+1 to Step004_BGPSL6-9-L30-F7+1
    using left_ur5_arm
                        -> Let go off Step003_BGPSL6-9-L30-F7+1 with right_ur5_arm
      *LET GO      -> Left Arm Step004_SLBNR6+1
```

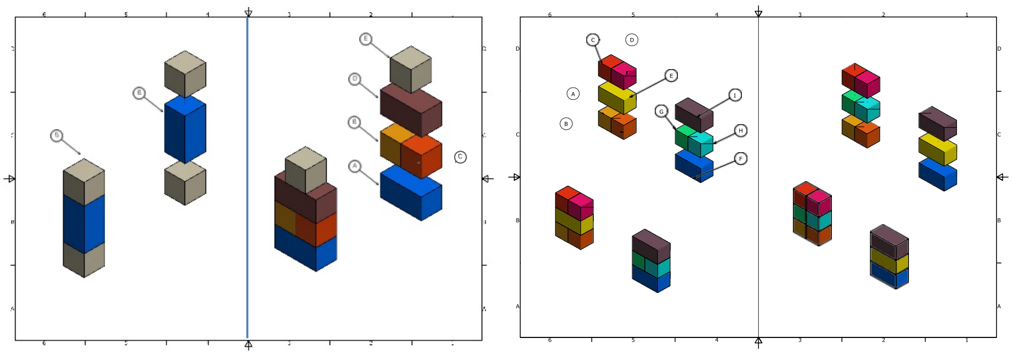
4.9 Skill execution and validation

Once the optimal assembly plan is generated as shown above, the software framework invokes the robotic manipulator to perform the various assembly tasks required to complete the assembly process. It should always be ensured that the assembly model is build based on the rules list described in the previous chapter. If the rules list is not followed the planning system will fail resulting in an unsupported assembly plan. The planning system works perfectly for the WRS belt drive assembly, as can be seen and verified in the plan. It should be further tested to include use cases that are different from the current assembly.

5 Use cases

This chapter describes the implementation of the developed sequence planner discussed in Chapter 3.3.2. The extracted data described in Section 3.1.3 is used to build an assembly plan as shown in the following use cases. The method has been expanded to include as many different use cases as possible.

5.1 Use case I: Box world



(a) (b) (c)

Figure 5.1: Assembly diagram for box world assembly. (a) Box world I containing three different blocks. (b) Box world II containing five different blocks. (c) Box world III containing nine different blocks. Source: Own illustration generated using Autodesk Inventor.

The first use case that we consider consists of a block world assembly. Three very simple examples are considered to test our developed model to verify if the assembly planner executes the assembly steps without any errors.

5.1.1 Box world I

The first example consists of three basic blocks and they are simply stacked one over the other.

Assembly sequence plan box world I

```
   SKILL SKIP      00: Grab Step001_Small+1 and insert into ground
   SKILL EXECUTION 01: Grab Step001_Big+1 and insert into Step001_Small+1 using
    left_ur5_arm
   SKILL EXECUTION 02: Grab Step001_Small+2 and insert into Step001_Big+1 using
    right_ur5_arm
```

5.1.2 Box world II

The second example consists of a slightly harder box world problem where up to five blocks are stacked over each other with the last box being placed in the middle of the whole stack.

Assembly sequence plan box world II

```
SKILL SKIP      00: Grab Step001_A and insert into ground
SKILL EXECUTION 01: Grab Step001_B and insert into Step001_A using left_ur5_arm
SKILL EXECUTION 02: Grab Step001_C and insert into Step001_A using right_ur5_arm
SKILL EXECUTION 03: Grab Step001_D and insert into Step001_C using left_ur5_arm
SKILL EXECUTION 04: Grab Step001_E and insert into Step001_D using right_ur5_arm
```

5.1.3 Box world III

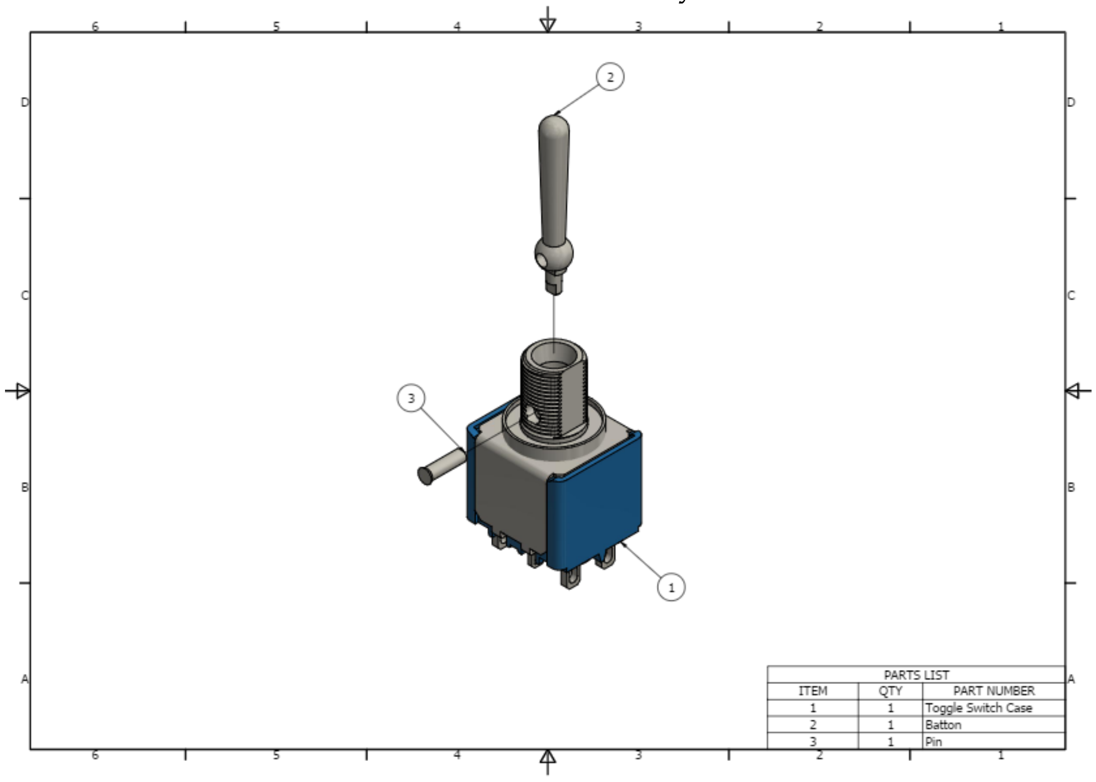
The third example consists of a rearranging problem were two nine different blocks are arranged in two different stacks and later the middle blocks between the two respective stacks are interchanged.

Assembly sequence plan box world III

```
SKILL SKIP      00: Grab Step001_A and insert into ground
SKILL EXECUTION 01: Grab Step001_B and insert into Step001_A using left_ur5_arm
SKILL EXECUTION 02: Grab Step001_E and insert into Step001_B using right_ur5_arm
SKILL EXECUTION 03: Grab Step001_D and insert into Step001_E using left_ur5_arm
SKILL EXECUTION 04: Grab Step001_C and insert into Step001_E using right_ur5_arm
SKILL EXECUTION 05: Grab Step001_G and insert into Step001_F using left_ur5_arm
SKILL EXECUTION 06: Grab Step001_H and insert into Step001_F using left_ur5_arm
SKILL EXECUTION 07: Grab Step001_I and insert into Step001_H using right_ur5_arm
SKILL UNDO 1ROT 08: Grab Step001_I and extract from Step001_H using right_ur5_arm
SKILL UNDO 1ROT 09: Grab Step001_H and extract from Step001_F using left_ur5_arm
SKILL UNDO 1ROT 10: Grab Step001_G and extract from Step001_F using left_ur5_arm
SKILL UNDO 1ROT 11: Grab Step001_C and extract from Step001_E using left_ur5_arm
SKILL UNDO 1ROT 12: Grab Step001_D and extract from Step001_E using left_ur5_arm
SKILL UNDO 1ROT 13: Grab Step001_E and extract from Step001_B using right_ur5_arm
SKILL EXECUTION 14: Grab Step002_G and insert into Step002_A using right_ur5_arm
SKILL EXECUTION 15: Grab Step002_H and insert into Step002_B using right_ur5_arm
SKILL EXECUTION 16: Grab Step002_H and insert into Step002_D using right_ur5_arm
SKILL EXECUTION 17: Grab Step002_G and insert into Step002_C using right_ur5_arm
SKILL EXECUTION 18: Grab Step002_F and insert into Step002_E using right_ur5_arm
SKILL EXECUTION 19: Grab Step002_I and insert into Step002_E using right_ur5_arm
```

5.2 Use case II: Toggle switch

The second use case consists of a toggle switch that contains three different parts. The challenge in this assembly is that if part #3 which is the pin is assembled before the baton, the whole assembly will fail.



PARTS LIST		
ITEM	QTY	PART NUMBER
1	1	Toggle Switch Case
2	1	Batton
3	1	Pin

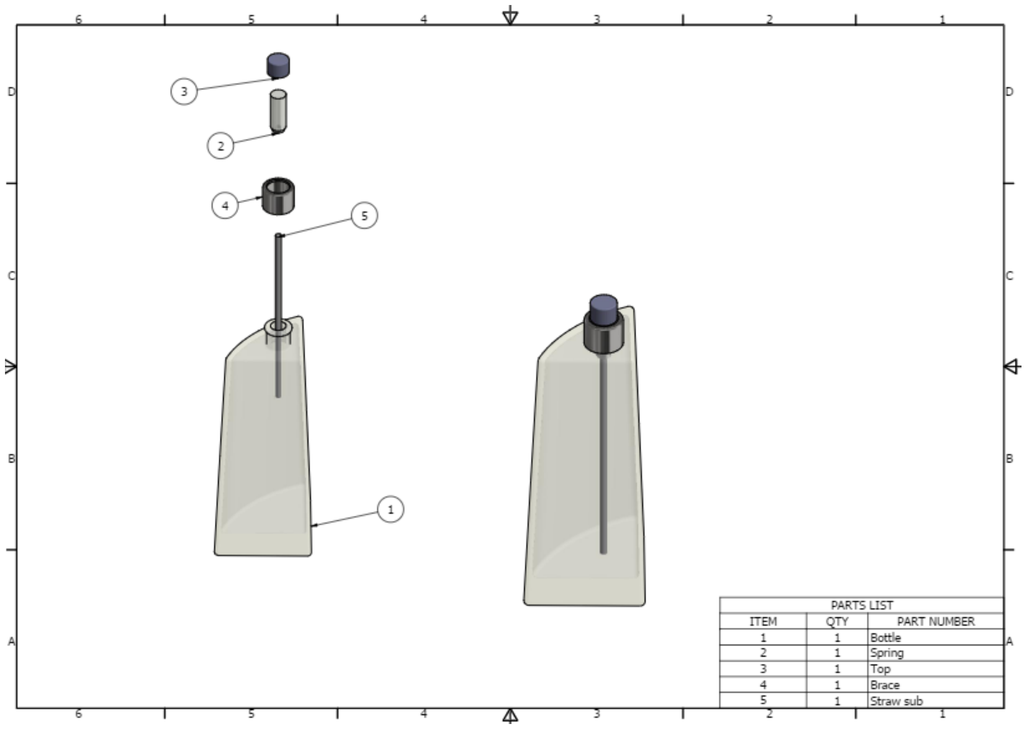
Figure 5.2: Assembly diagram with parts list for toggle switch, Source: Own illustration generated using Autodesk Inventor.

Assembly sequence plan toggle switch

```
00: Grab Toggle-Switch-Case and insert into ground
01: Grab Batton and insert into Toggle-Switch-Case using L_Arm_UR5
02: Grab Pin and insert into Batton using R_Arm_UR5
```

5.3 Use case III: Perfume bottle

The third use case consists of a perfume bottle assembly. The challenge in this assembly is that the straw should be placed inside the bottle before the rest of the components are assembled.



PARTS LIST		
ITEM	QTY	PART NUMBER
1	1	Bottle
2	1	Spring
3	1	Top
4	1	Brace
5	1	Straw sub

Figure 5.3: Assembly diagram with parts list for perfume bottle, Source: Own illustration generated using Autodesk Inventor.

Assembly sequence plan perfume bottle

```
01:    Grab _Spring and screw into _Straw-sub using L_Arm_UR5
SKIP   01: Grab _Bottle and screw into ground
02:    Grab _Spring and screw into _Bottle using L_Arm_UR5
03:    Grab _Top and screw into _Spring using L_Arm_UR5
04:    Grab _Brace and screw into _Top using L_Arm_UR5
```

5.4 Use case IV: Clutch assembly

The fourth use case consists of a clutch plate assembly. The main characteristic of this assembly is that it is a stacking assembly. The main challenge in this assembly is that it consists of a sub assembly and a main assembly. So, in order to completely assemble the clutch the sub-assembly must be assembled first before the main assembly is put together.

Assembly sequence plan: main assembly of clutch plate

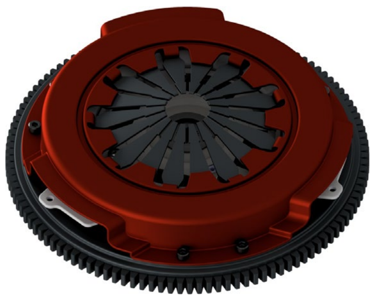
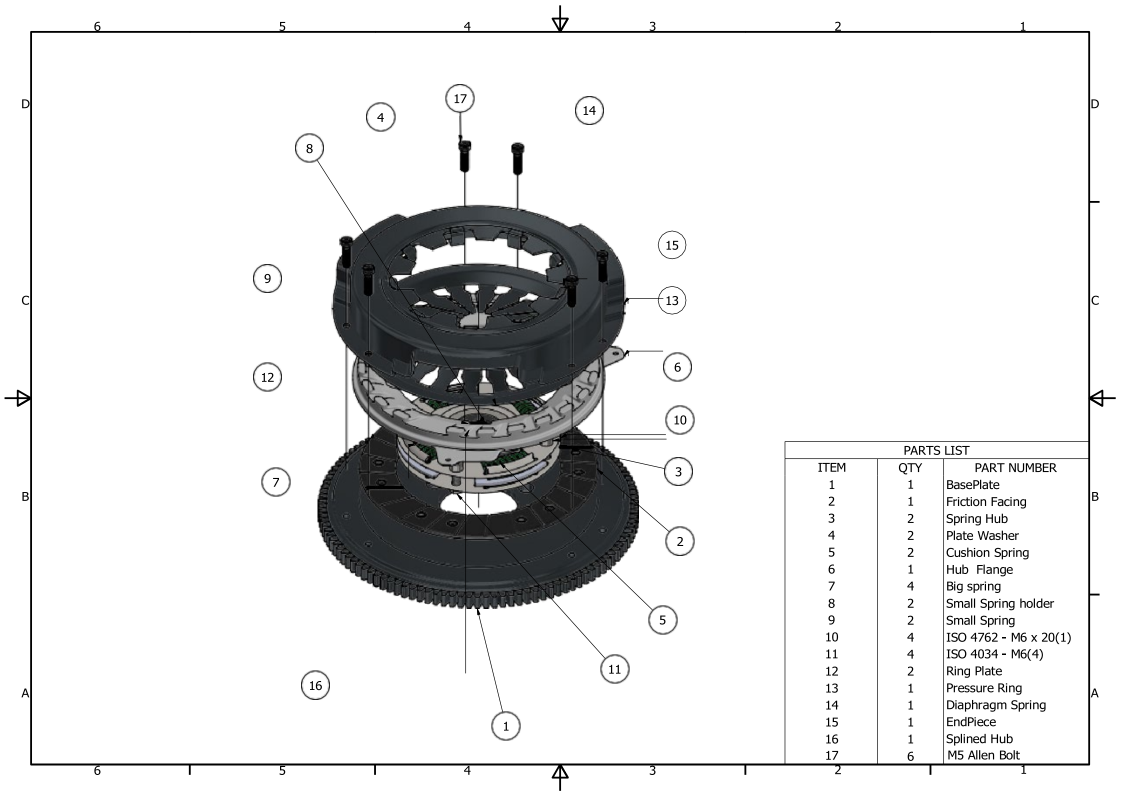


Figure 5.4: An assembled clutch, Source: Own illustration generated using Autodesk Inventor.

```
Grab BasePlate and insert into ground
Grab Friction-Facing and insert into BasePlate using L_Arm_UR5
Grab Internal-sub-assembly and insert into Friction-Facing using R_Arm_UR5
Grab Ring-Plate and insert into Internal-sub-assembly using L_Arm_UR5
Grab Ring-Plate and insert into Internal-sub-assembly using R_Arm_UR5
Grab Pressure-Ring and insert into Internal-sub-assembly using L_Arm_UR5
Grab Splined-Hub and insert into Internal-sub-assembly using R_Arm_UR5
Grab Diaphragm-Spring and insert into EndPiece using L_Arm_UR5
Grab M5-Allen-Bolt and screw into EndPiece using L_Arm_UR5
Grab M5-Allen-Bolt and screw into EndPiece using L_Arm_UR5
Grab M5-Allen-Bolt and screw into EndPiece using L_Arm_UR5
Grab M5-Allen-Bolt and screw into EndPiece using L_Arm_UR5
Grab M5-Allen-Bolt and screw into EndPiece using L_Arm_UR5
Grab M5-Allen-Bolt and screw into EndPiece using L_Arm_UR5
Grab Diaphragm-Spring and insert into Pressure-Ring using L_Arm_UR5
```

Assembly sequence plan: sub assembly of clutch plate

```
Grab Spring-Hub and insert into ground
Grab Plate-Washer and insert into Spring-Hub using L_Arm_UR5
Grab Cushion-Spring and insert into Plate-Washer using R_Arm_UR5
Grab Hub-Flange and insert into Cushion-Spring using L_Arm_UR5
Grab Cushion-Spring and insert into Hub-Flange using R_Arm_UR5
Grab Big-spring and insert into Hub-Flange using L_Arm_UR5
Grab Big-spring and insert into Hub-Flange using R_Arm_UR5
Grab Big-spring and insert into Hub-Flange using L_Arm_UR5
Grab Big-spring and insert into Hub-Flange using R_Arm_UR5
Grab Plate-Washer and insert into Cushion-Spring using L_Arm_UR5
Grab Spring-Hub and screw into Plate-Washer using R_Arm_UR5
Grab ISO-4762-M6-x-20 and screw into Spring-Hub using R_Arm_UR5
Grab ISO-4762-M6-x-20 and screw into Spring-Hub using R_Arm_UR5
Grab ISO-4762-M6-x-20 and screw into Spring-Hub using R_Arm_UR5
Grab ISO-4762-M6-x-20 and screw into Spring-Hub using R_Arm_UR5
Grab ISO-4034-M6 and screw into ISO-4762-M6-x-20 using R_Arm_UR5
Grab ISO-4034-M6 and screw into ISO-4762-M6-x-20 using R_Arm_UR5
Grab ISO-4034-M6 and screw into ISO-4762-M6-x-20 using R_Arm_UR5
Grab ISO-4034-M6 and screw into ISO-4762-M6-x-20 using R_Arm_UR5
```



PARTS LIST		
ITEM	QTY	PART NUMBER
1	1	BasePlate
2	1	Friction Facing
3	2	Spring Hub
4	2	Plate Washer
5	2	Cushion Spring
6	1	Hub Flange
7	4	Big spring
8	2	Small Spring holder
9	2	Small Spring
10	4	ISO 4762 - M6 x 20(1)
11	4	ISO 4034 - M6(4)
12	2	Ring Plate
13	1	Pressure Ring
14	1	Diaphragm Spring
15	1	EndPiece
16	1	Splined Hub
17	6	M5 Allen Bolt

Figure 5.5: Assembly diagram with parts list for clutch plate, Source: Own illustration generated using Autodesk Inventor.

6 Summary and results

This section talks about the overall summary of the thesis and the results that were realised from the method section.

6.1 Summary

The main focus of the thesis was to make a state-of-the-art autonomous assembly system. Chapter 1 provides an overview of the background that led to the thesis. It also talks about the objectives and defines an approach to tackle the problem statement.

The theoretical framework necessary for implementing such a system is addressed in Chapter 2. The assembly workflow in an autonomous assembly system is visualized and the assembly sequence is defined. It is followed by defining the essential elements of the robotic system. Lastly, the system architecture and the software framework of the autonomous system is specified.

Chapter 3 talks about the method used to build the system. The theoretical aspects that were identified from the previous chapter are further expanded by studying the different data representation models in CAD. It also defines the assembly rules that must be followed when making a CAD model. If the defined rules are not followed the planning system will fail and no optimum assembly plan will be generated.

Chapter 4 talks about the implementation and validation of the proposed system mentioned in the previous chapter. The data set that needs to be extracted is defined and a strategy to generate an assembly sequence plan is outlined.

In Chapter 5, different use cases are identified and tested with the newly developed system so that distinct corner cases are covered. The system is tested with a clutch plate, a perfume bottle, a toggle switch, and a block assembly.

6.2 Results

The methodology developed through this thesis has been used to create an autonomous assembly planning system. The WRS belt drive unit along with four other use cases was used to test and validate the developed framework. Autodesk Inventor was chosen as the proposed CAD system that provided the required data sets.

The developed framework identifies components and subassemblies that make up the final product. Autodesk Inventor API was used to identify and extract the required data points. All the components and sub-assemblies of the required final assembly were identified along with their respective constraints, poses and orientations.

An Inventor assembly exporter was also developed to generate the assembly graph that visually represents the correlation between different parts and subassemblies. The assembly plan was generated using the assembly graph and by calculating the interrelationships between each component.

The generated assembly plan is then used to execute the skills or the primitives by the robotic manipulators to assemble the final product or the assembly. The test cases confirm that the developed model can generate the correct assembly plan for all the different test cases that were tested with the system.

7 Discussion

While analysing the literature research about the WRS, it was found that none of the teams has developed a system that was able to build an assembly plan on the fly. Every single team that competed had a significant change over time. The thesis aimed at finding a solution that could automatically extract the required planning data needed for the robotic reasoner. The analysis of the results from the thesis indicates that it was possible to build an autonomous robotic system that was able to build an assembly plan from the CAD design.

The results suggest that the use of a proprietary CAD system enabled the easiest method for the extraction of assembly data as opposed to the findings of the study proposed by Za and Du using a STEP based method.[70] The proposed system did not require an additional subsystem that pulled out the geometrical, physical, and mechanical constraints.

The result also aligns with the findings of Mathew and Rao which proposed a method to generate the liaison relationships from an assembly using commercial CAD software.[71] The currently developed system even expands further on the method of using API to generate liaison relationship. The system developed by Mathew and Rao has only the possibility to generate the snapshot of the relational database and the liaison table. The currently developed system on the other hand can generate the relationship graph and the assembly plan and also execute the required primitive skills required to perform the complete assembly by the robotic manipulators.

The previous research of Chaudron, Martin, and Godot proposed a method that uses the "And/Or Graph" to plan the assembly sequence operations. The method was implemented using a proprietary CAD system; Catia V5.[72] The results from the thesis was in line with the findings of the above-mentioned study.

[70] Conf. Zha & Du, *supra* note 52, p. 1087-110.

[71] Conf. Mathew & Rao (2010), p. 167-75.

[72] Conf. Chaudron, Martin & Godot (2005), p. 156-61.

The generated assembly sequence plans can be also be used on shop floors for manual assembly after extensive testing of the system. The main advantage lies in the fact that the operator of the proposed system does not need any knowledge to generate the plan.

The assembly plans in the form of step-by-step nodal graphs also give so much clarity to the assembly line worker, unlike other engineering drawings. The flexibility to generate text outputs, graphs as well as parts list makes it easier for the factory line worker to understand the production processes faster. When it comes to the autonomous system, the plans that are developed on the go will provide most assembly lines with agility and leanness thereby helping to make quick product changeovers in the shortest span possible.

The main advantage of the system compared to previously existing systems is that it does not require any user-generated input to complete the assembly process, as long as the rules specified inside the rule box is followed.

The current developed system currently has some limitations. The plans generated are sometimes not optimal or there might be special corner cases that would appear as more and more test cases are performed on the system. With further development, the system will be able to achieve optimal solutions as more and more corner cases get integrated into the system. Another limitation arises from the use of a proprietary CAD system and thus the user is bound to the product ecosystem.

8 Future scope

In this thesis, a rigorous theoretical framework for data extraction from a proprietary CAD system was proposed. The framework allows extraction of geometric and user-defined data sets from an assembly drawing.

There are many directions for future work which are briefly summarised next.

- The vision system can be expanded to detect graspable points from cluttered scenes and picking bins. Since the current pose is given from the CAD data, the future system can use various sensors and camera systems to detect object surface states, sizes and more.
- Human-Robot Collaborative assembly tasks that effectively collaborate with the human workers can be considered. Chandra et al. proposed a robotic platform that performs ‘picking’, ‘showing’, ‘placing’ and ‘handover’ actions on real-world objects in coordination with the human.[73] Instead of the current framework a cognitive architecture can be developed to reason and plan the human-robot collaboration scenario.
- Another possible direction of research is a hybrid manufacturing cell. Sadrfaridpour et al. propose an integrated physical and social human-robot collaboration (HRC) framework for assembly tasks in a hybrid manufacturing cell.[74] A computational model can be developed to integrate the HRC framework in future research.
- Another research direction builds on already existing knowledge that is available and converting the structured data into an RDF-based knowledge base.[75] This knowledge is then gradually extended during the definition of the robot task.

[73] Conf. Chandra et al. (2016), p. 3-14.

[74] Conf. Sadrfaridpour, Saeidi & Wang (82016), p. 462-67.

[75] Conf. Bjorkelund et al. (2011).

10 Conclusion

This research aimed at the autonomous assembly of Industrial robots and to extract the data that is required to do the task planning. Based on the literature research, practical implementation and the analysis of the four different test cases, it can be concluded that the required data can be extracted from the system using the proposed system. The proposed system can also be scaled to include industrial applications and as more corner cases are tested and more data is added into the OWL file, the system can span to include multiple other use cases.

The research also clearly illustrates that any geometric, as well as semantic data that is embedded into the design drawing, can be extracted and reused to include other phases of the manufacturing process. But it also raises the question of building the CAD assembly based on a predefined set of rules. The definition of a predefined set of rules can become sometimes confusing to a new user.

The study was able to solve the problem of autonomously planning the assembly tasks from the extracted data models and execute the required skills to complete the final assembly.

Based on these conclusions the future researchers can aim to address the problems like eliminating the use of predefined rules. Future studies could also aim at an approach that can integrate human-robot collaborative assembly tasks.

Appendix A

Use case I: Box world I

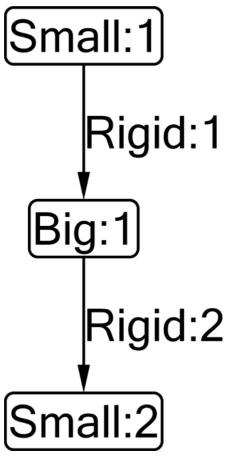


Figure .1: Detailed relationship node diagram for box world I,
Source: Own illustration generated using Inventor assembly exporter.

```
SKILL SKIP      00: Grab Step001_Small+1 and insert into ground
SKILL EXECUTION 01: Grab Step001_Big+1 and insert into Step001_Small+1 using
left_ur5_arm
                 -> Grab Step001_Big+1 using left_ur5_arm
                 -> Insert Step001_Big+1 into Step001_Small+1 using
left_ur5_arm
SKILL EXECUTION 02: Grab Step001_Small+2 and insert into Step001_Big+1 using
right_ur5_arm
                 -> Grab Step001_Small+2 using right_ur5_arm
                 -> Insert Step001_Small+2 into Step001_Big+1 using
right_ur5_arm
                 -> Let go off Step001_Big+1 with left_ur5_arm
                 -> Let go off Step001_Small+2 with right_ur5_arm
```

Use case 1: Box world II

```
SKILL SKIP      00: Grab Step001_A and insert into ground
SKILL EXECUTION 01: Grab Step001_B and insert into Step001_A using
left_ur5_arm
                 -> Grab Step001_B using left_ur5_arm
                 -> Insert Step001_B into Step001_A using left_ur5_arm
SKILL EXECUTION 02: Grab Step001_C and insert into Step001_A using
right_ur5_arm
                 -> Grab Step001_C using right_ur5_arm
                 -> Insert Step001_C into Step001_A using right_ur5_arm
                 -> Let go off Step001_B with left_ur5_arm
SKILL EXECUTION 03: Grab Step001_D and insert into Step001_C using
left_ur5_arm
                 -> Grab Step001_D using left_ur5_arm
                 -> Insert Step001_D into Step001_C using left_ur5_arm
                 -> Let go off Step001_C with right_ur5_arm
SKILL EXECUTION 04: Grab Step001_E and insert into Step001_D using
right_ur5_arm
```

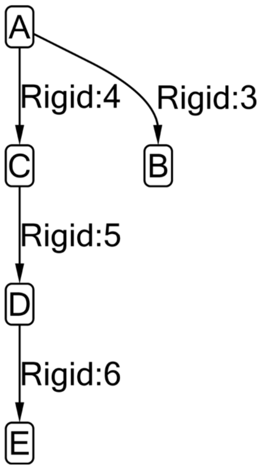


Figure .2: Detailed relationship node diagram for box world II,
Source: Own illustration generated using Inventor assembly exporter.

```
                            -> Grab Step001_E using right_ur5_arm
                            -> Insert Step001_E into Step001_D using right_ur5_arm
                            -> Let go off Step001_D with left_ur5_arm
                            -> Let go off Step001_E with right_ur5_arm
```

Use case I: Box world III

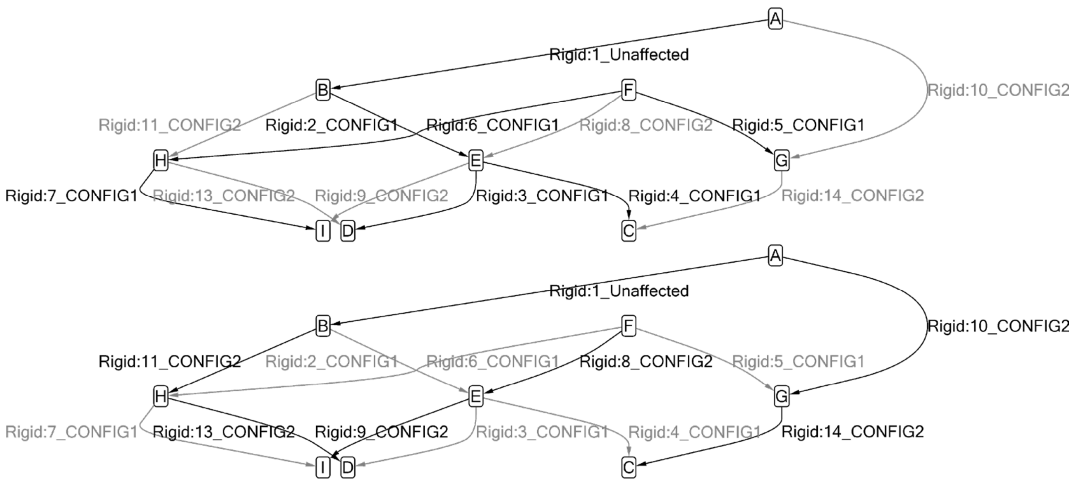


Figure .3: Detailed relationship node diagram for box world III,
Source: Own illustration generated using Inventor assembly exporter.

```
    SKILL SKIP      00: Grab Step001_A and insert into ground
    SKILL EXECUTION 01: Grab Step001_B and insert into Step001_A using
    left_ur5_arm
                        -> Grab Step001_B using left_ur5_arm
```

```
                    -> Insert Step001_B into Step001_A using left_ur5_arm
SKILL EXECUTION 02: Grab Step001_E and insert into Step001_B using
right_ur5_arm
                    -> Grab Step001_E using right_ur5_arm
                    -> Insert Step001_E into Step001_B using right_ur5_arm
                    -> Let go off Step001_B with left_ur5_arm
SKILL EXECUTION 03: Grab Step001_D and insert into Step001_E using
left_ur5_arm
                    -> Grab Step001_D using left_ur5_arm
                    -> Insert Step001_D into Step001_E using left_ur5_arm
                    -> Let go off Step001_E with right_ur5_arm
SKILL EXECUTION 04: Grab Step001_C and insert into Step001_E using
right_ur5_arm
                    -> Grab Step001_C using right_ur5_arm
                    -> Insert Step001_C into Step001_E using right_ur5_arm
                    -> Let go off Step001_D with left_ur5_arm
                    -> Let go off Step001_C with right_ur5_arm
SKILL EXECUTION 05: Grab Step001_G and insert into Step001_F using
left_ur5_arm
                    -> Grab Step001_G using left_ur5_arm
                    -> Grab Step001_F using right_ur5_arm
                    -> Insert Step001_G into Step001_F using left_ur5_arm
                    -> Let go off Step001_G with left_ur5_arm
SKILL EXECUTION 06: Grab Step001_H and insert into Step001_F using
left_ur5_arm
                    -> Grab Step001_H using left_ur5_arm
                    -> Insert Step001_H into Step001_F using left_ur5_arm
SKILL EXECUTION 07: Grab Step001_I and insert into Step001_H using
right_ur5_arm
                    -> Grab Step001_I using right_ur5_arm
                    -> Insert Step001_I into Step001_H using right_ur5_arm
                    -> Let go off Step001_I with right_ur5_arm
SKILL UNDO 1ROT 08: Grab Step001_I and extract from Step001_H using
right_ur5_arm
                    -> Grab Step001_I using right_ur5_arm
                    -> Grab Step001_I using right_ur5_arm
                    -> Let go off Step001_I with right_ur5_arm
SKILL UNDO 1ROT 09: Grab Step001_H and extract from Step001_F using
left_ur5_arm
                    -> Grab Step001_F using right_ur5_arm
                    -> Grab Step001_H using left_ur5_arm
                    -> Let go off Step001_H with left_ur5_arm
SKILL UNDO 1ROT 10: Grab Step001_G and extract from Step001_F using
left_ur5_arm
                    -> Grab Step001_G using left_ur5_arm
                    -> Grab Step001_G using left_ur5_arm
                    -> Let go off Step001_G with left_ur5_arm
SKILL UNDO 1ROT 11: Grab Step001_C and extract from Step001_E using
left_ur5_arm
                    -> Grab Step001_C using left_ur5_arm
                    -> Grab Step001_E using right_ur5_arm
                    -> Grab Step001_C using left_ur5_arm
                    -> Let go off Step001_C with left_ur5_arm
SKILL UNDO 1ROT 12: Grab Step001_D and extract from Step001_E using
left_ur5_arm
                    -> Grab Step001_D using left_ur5_arm
                    -> Grab Step001_D using left_ur5_arm
                    -> Let go off Step001_D with left_ur5_arm
SKILL UNDO 1ROT 13: Grab Step001_E and extract from Step001_B using
right_ur5_arm
                    -> Grab Step001_B using left_ur5_arm
                    -> Grab Step001_E using right_ur5_arm
```

```
                         -> Let go off Step001_E with right_ur5_arm
    SKILL EXECUTION 14: Grab Step002_G and insert into Step002_A using
    right_ur5_arm
                         -> Grab Step002_G using right_ur5_arm
                         -> Insert Step002_G into Step002_A using right_ur5_arm
                         -> Let go off Step002_G with right_ur5_arm
    SKILL EXECUTION 15: Grab Step002_H and insert into Step002_B using
    right_ur5_arm
                         -> Grab Step002_H using right_ur5_arm
                         -> Insert Step002_H into Step002_B using right_ur5_arm
    SKILL EXECUTION 16: Grab Step002_H and insert into Step002_D using
    right_ur5_arm
                         -> Insert Step002_H into Step002_D using right_ur5_arm
                         -> Let go off Step002_H with right_ur5_arm
    SKILL EXECUTION 17: Grab Step002_G and insert into Step002_C using
    right_ur5_arm
                         -> Grab Step002_G using right_ur5_arm
                         -> Insert Step002_G into Step002_C using right_ur5_arm
                         -> Let go off Step002_G with right_ur5_arm
    SKILL EXECUTION 18: Grab Step002_F and insert into Step002_E using
    right_ur5_arm
                         -> Grab Step002_F using right_ur5_arm
                         -> Insert Step002_F into Step002_E using right_ur5_arm
                         -> Let go off Step002_F with right_ur5_arm
    SKILL EXECUTION 19: Grab Step002_I and insert into Step002_E using
    right_ur5_arm
                         -> Grab Step002_I using right_ur5_arm
                         -> Insert Step002_I into Step002_E using right_ur5_arm
                         -> Let go off Step001_B with left_ur5_arm
                         -> Let go off Step002_I with right_ur5_arm
```

Use case II: Toggle Switch

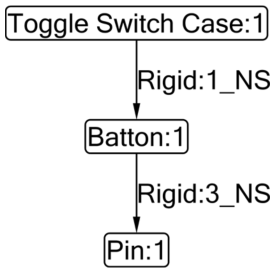


Figure .4: Detailed relationship node diagram for toggle switch,
Source: Own illustration generated using Inventor assembly exporter.

```
    SKILL SKIP      00: Grab Step001_Toggle-Switch-Case+1 and insert into ground
    SKILL EXECUTION 01: Grab Step001_Batton+1 and insert into Step001_Toggle-
    Switch-Case+1 using left_ur5_arm
                         -> Grab Step001_Batton+1 using left_ur5_arm
                         -> Insert Step001_Batton+1 into Step001_Toggle-Switch-Case+1
    using left_ur5_arm
    SKILL EXECUTION 02: Grab Step001_Pin+1 and insert into Step001_Batton+1 using
    right_ur5_arm
```

```
                      -> Grab Step001_Pin+1 using right_ur5_arm
                      -> Insert Step001_Pin+1 into Step001_Batton+1 using
    right_ur5_arm
                      -> Let go off Step001_Batton+1 with left_ur5_arm
                      -> Let go off Step001_Pin+1 with right_ur5_arm
```

Use case III: Perfume bottle

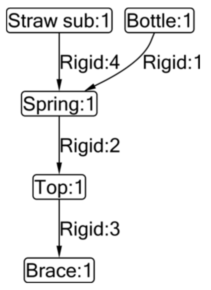


Figure .5: Detailed relationship node diagram for perfume bottle,
Source: Own illustration generated using Inventor assembly exporter

```
    SKILL EXECUTION 01: Grab Step001_Spring+1 and screw into Step001_Straw-sub+1
    using left_ur5_arm
    -> Grab Step001_Spring+1 using left_ur5_arm
    -> Grab Step001_Straw-sub+1 using right_ur5_arm
    -> Screw Step001_Spring+1 to Step001_Straw-sub+1 using left_ur5_arm
SKILL SKIP      01: Grab Step001_Bottle+1 and screw into ground
SKILL EXECUTION 02: Grab Step001_Spring+1 and screw into Step001_Bottle+1 using
    left_ur5_arm
    -> Grab Step001_Spring+1 using left_ur5_arm
    -> Screw Step001_Spring+1 to Step001_Bottle+1 using left_ur5_arm
SKILL EXECUTION 03: Grab Step001_Top+1 and screw into Step001_Spring+1 using
    left_ur5_arm
    -> Grab Step001_Top+1 using left_ur5_arm
    -> Screw Step001_Top+1 to Step001_Spring+1 using left_ur5_arm
SKILL EXECUTION 04: Grab Step001_Brace+1 and screw into Step001_Top+1 using
    left_ur5_arm
    -> Grab Step001_Brace+1 using left_ur5_arm
    -> Screw Step001_Brace+1 to Step001_Top+1 using left_ur5_arm
    -> Let go off Step001_Straw-sub+1 with right_ur5_arm
```

Use case IV: Clutch assembly

```
    00: Grab Spring-Hub and insert into ground
    01: Grab Plate-Washer and insert into Spring-Hub using L_Arm_UR5
        -> Grab Plate-Washer using L_Arm_UR5
        -> Insert Plate-Washer into Spring-Hub using L_Arm_UR5
    02: Grab Cushion-Spring and insert into Plate-Washer using R_Arm_UR5
        -> Grab Cushion-Spring using R_Arm_UR5
        -> Insert Cushion-Spring into Plate-Washer using R_Arm_UR5
```

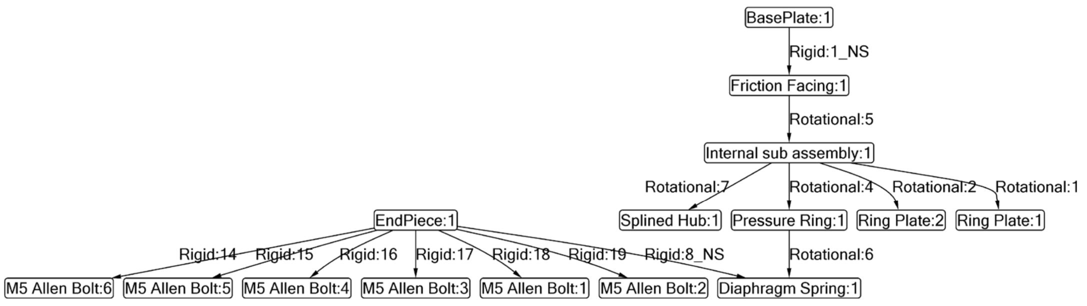


Figure .6: Detailed relationship node diagram for clutch assembly, Source: Own illustration generated using Inventor assembly exporter

```
        -> Let go off Plate-Washer with L_Arm_UR5
    03: Grab Hub-Flange and insert into Cushion-Spring using L_Arm_UR5
        -> Grab Hub-Flange using L_Arm_UR5
        -> Insert Hub-Flange into Cushion-Spring using L_Arm_UR5
        -> Let go off Cushion-Spring with R_Arm_UR5
    04: Grab Cushion-Spring and insert into Hub-Flange using R_Arm_UR5
        -> Grab Cushion-Spring using R_Arm_UR5
        -> Insert Cushion-Spring into Hub-Flange using R_Arm_UR5
        -> Let go off Hub-Flange with L_Arm_UR5
    05: Grab Big-spring and insert into Hub-Flange using L_Arm_UR5
        -> Grab Big-spring using L_Arm_UR5
        -> Insert Big-spring into Hub-Flange using L_Arm_UR5
        -> Let go off Cushion-Spring with R_Arm_UR5
    06: Grab Big-spring and insert into Hub-Flange using R_Arm_UR5
        -> Grab Big-spring using R_Arm_UR5
        -> Insert Big-spring into Hub-Flange using R_Arm_UR5
        -> Let go off Big-spring with L_Arm_UR5
    07: Grab Big-spring and insert into Hub-Flange using L_Arm_UR5
        -> Grab Big-spring using L_Arm_UR5
        -> Insert Big-spring into Hub-Flange using L_Arm_UR5
        -> Let go off Big-spring with R_Arm_UR5
    08: Grab Big-spring and insert into Hub-Flange using R_Arm_UR5
        -> Grab Big-spring using R_Arm_UR5
        -> Insert Big-spring into Hub-Flange using R_Arm_UR5
        -> Let go off Big-spring with L_Arm_UR5
    09: Grab Plate-Washer and insert into Cushion-Spring using L_Arm_UR5
        -> Grab Plate-Washer using L_Arm_UR5
        -> Insert Plate-Washer into Cushion-Spring using L_Arm_UR5
        -> Let go off Big-spring with R_Arm_UR5
    10: Grab Spring-Hub and screw into Plate-Washer using R_Arm_UR5
        -> Grab Spring-Hub using R_Arm_UR5
        -> Screw Spring-Hub to Plate-Washer using R_Arm_UR5
    11: Grab ISO-4762-M6-x-20 and screw into Spring-Hub using R_Arm_UR5
        -> Grab ISO-4762-M6-x-20 using R_Arm_UR5
        -> Screw ISO-4762-M6-x-20 to Spring-Hub using R_Arm_UR5
    12: Grab ISO-4762-M6-x-20 and screw into Spring-Hub using R_Arm_UR5
        -> Grab ISO-4762-M6-x-20 using R_Arm_UR5
        -> Screw ISO-4762-M6-x-20 to Spring-Hub using R_Arm_UR5
    13: Grab ISO-4762-M6-x-20 and screw into Spring-Hub using R_Arm_UR5
        -> Grab ISO-4762-M6-x-20 using R_Arm_UR5
        -> Screw ISO-4762-M6-x-20 to Spring-Hub using R_Arm_UR5
    14: Grab ISO-4762-M6-x-20 and screw into Spring-Hub using R_Arm_UR5
        -> Grab ISO-4762-M6-x-20 using R_Arm_UR5
        -> Screw ISO-4762-M6-x-20 to Spring-Hub using R_Arm_UR5
    15: Grab ISO-4034-M6 and screw into ISO-4762-M6-x-20 using R_Arm_UR5
```

```
        -> Grab ISO-4034-M6 using R_Arm_UR5
        -> Screw ISO-4034-M6 to ISO-4762-M6-x-20 using R_Arm_UR5
    16: Grab ISO-4034-M6 and screw into ISO-4762-M6-x-20 using R_Arm_UR5
        -> Grab ISO-4034-M6 using R_Arm_UR5
        -> Screw ISO-4034-M6 to ISO-4762-M6-x-20 using R_Arm_UR5
    17: Grab ISO-4034-M6 and screw into ISO-4762-M6-x-20 using R_Arm_UR5
        -> Grab ISO-4034-M6 using R_Arm_UR5
        -> Screw ISO-4034-M6 to ISO-4762-M6-x-20 using R_Arm_UR5
    18: Grab ISO-4034-M6 and screw into ISO-4762-M6-x-20 using R_Arm_UR5
        -> Grab ISO-4034-M6 using R_Arm_UR5
        -> Screw ISO-4034-M6 to ISO-4762-M6-x-20 using R_Arm_UR5
        -> Let go off Plate-Washer with L_Arm_UR5
```

Clutch Subassembly

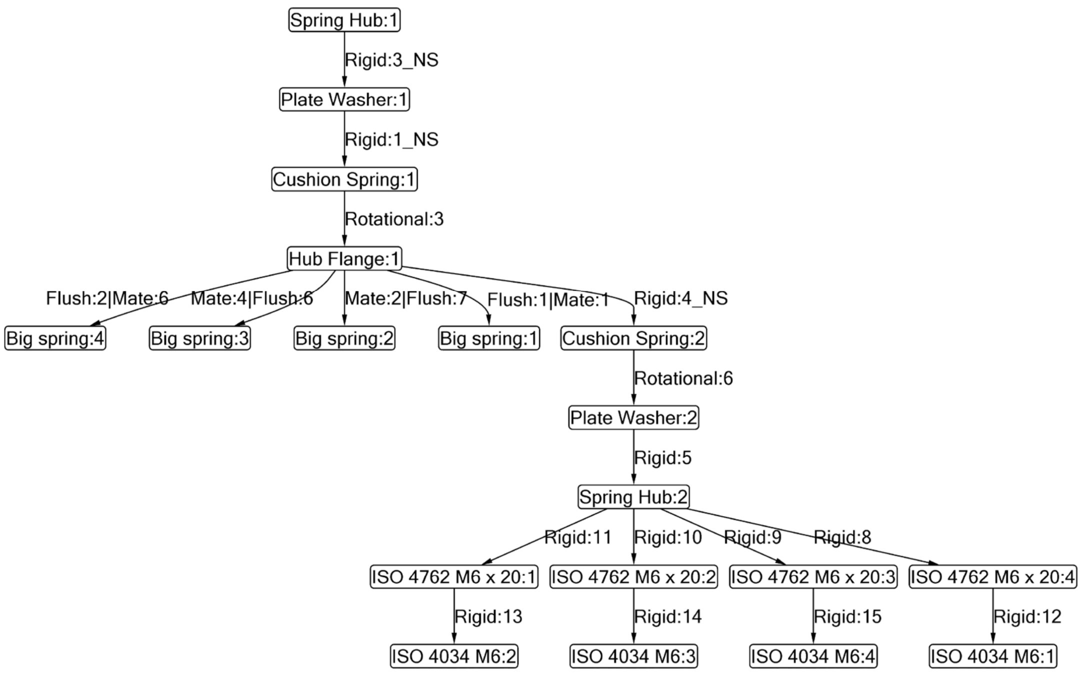


Figure .7: Detailed relationship node diagram for clutch sub assembly, Source: Own illustration generated using Inventor assembly exporter

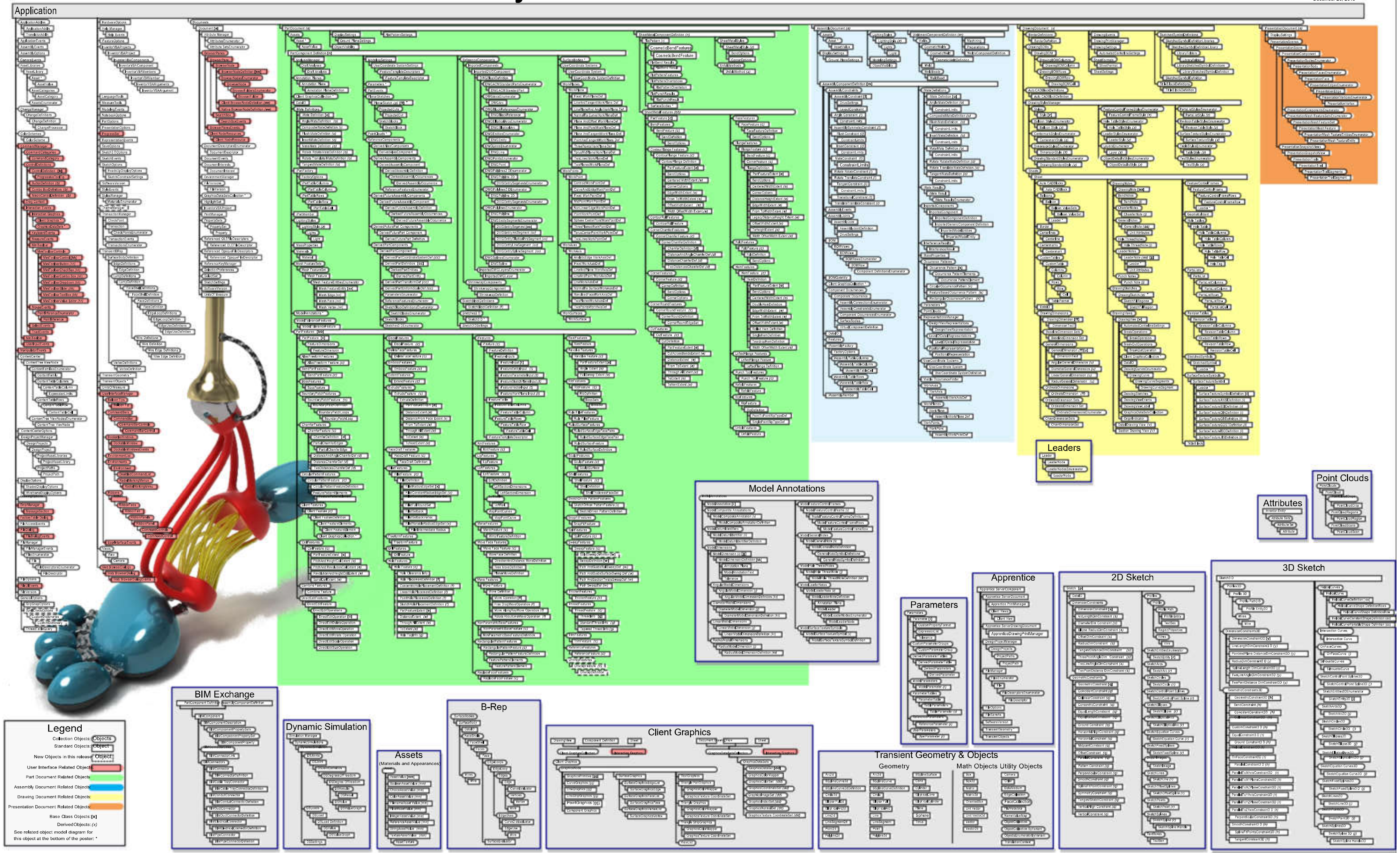
```
    00: Grab Spring-Hub and insert into ground
    01: Grab Plate-Washer and insert into Spring-Hub using L_Arm_UR5
        -> Grab Plate-Washer using L_Arm_UR5
        -> Insert Plate-Washer into Spring-Hub using L_Arm_UR5
    02: Grab Cushion-Spring and insert into Plate-Washer using R_Arm_UR5
        -> Grab Cushion-Spring using R_Arm_UR5
        -> Insert Cushion-Spring into Plate-Washer using R_Arm_UR5
        -> Let go off Plate-Washer with L_Arm_UR5
    03: Grab Hub-Flange and insert into Cushion-Spring using L_Arm_UR5
        -> Grab Hub-Flange using L_Arm_UR5
        -> Insert Hub-Flange into Cushion-Spring using L_Arm_UR5
        -> Let go off Cushion-Spring with R_Arm_UR5
    04: Grab Cushion-Spring and insert into Hub-Flange using R_Arm_UR5
```

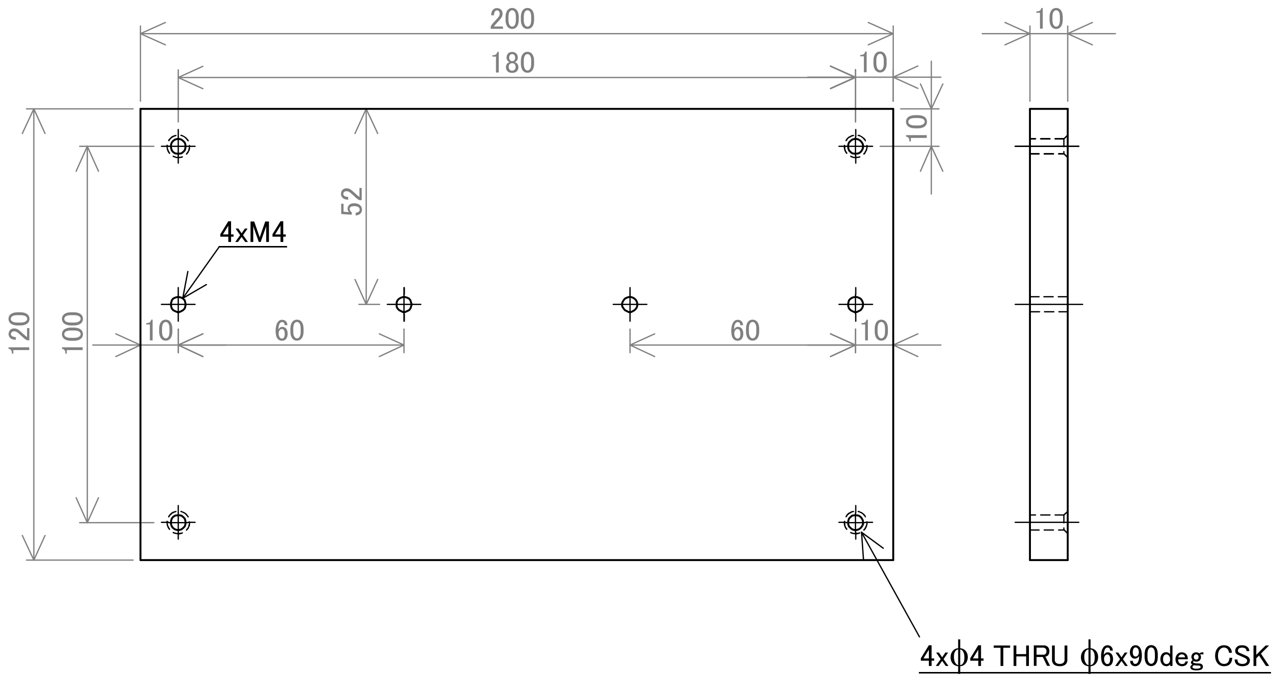
```
        -> Grab Cushion-Spring using R_Arm_UR5
        -> Insert Cushion-Spring into Hub-Flange using R_Arm_UR5
        -> Let go off Hub-Flange with L_Arm_UR5
    05: Grab Big-spring and insert into Hub-Flange using L_Arm_UR5
        -> Grab Big-spring using L_Arm_UR5
        -> Insert Big-spring into Hub-Flange using L_Arm_UR5
        -> Let go off Cushion-Spring with R_Arm_UR5
    06: Grab Big-spring and insert into Hub-Flange using R_Arm_UR5
        -> Grab Big-spring using R_Arm_UR5
        -> Insert Big-spring into Hub-Flange using R_Arm_UR5
        -> Let go off Big-spring with L_Arm_UR5
    07: Grab Big-spring and insert into Hub-Flange using L_Arm_UR5
        -> Grab Big-spring using L_Arm_UR5
        -> Insert Big-spring into Hub-Flange using L_Arm_UR5
        -> Let go off Big-spring with R_Arm_UR5
    08: Grab Big-spring and insert into Hub-Flange using R_Arm_UR5
        -> Grab Big-spring using R_Arm_UR5
        -> Insert Big-spring into Hub-Flange using R_Arm_UR5
        -> Let go off Big-spring with L_Arm_UR5
    09: Grab Plate-Washer and insert into Cushion-Spring using L_Arm_UR5
        -> Grab Plate-Washer using L_Arm_UR5
        -> Insert Plate-Washer into Cushion-Spring using L_Arm_UR5
        -> Let go off Big-spring with R_Arm_UR5
    10: Grab Spring-Hub and screw into Plate-Washer using R_Arm_UR5
        -> Grab Spring-Hub using R_Arm_UR5
        -> Screw Spring-Hub to Plate-Washer using R_Arm_UR5
    11: Grab ISO-4762-M6-x-20 and screw into Spring-Hub using R_Arm_UR5
        -> Grab ISO-4762-M6-x-20 using R_Arm_UR5
        -> Screw ISO-4762-M6-x-20 to Spring-Hub using R_Arm_UR5
    12: Grab ISO-4762-M6-x-20 and screw into Spring-Hub using R_Arm_UR5
        -> Grab ISO-4762-M6-x-20 using R_Arm_UR5
        -> Screw ISO-4762-M6-x-20 to Spring-Hub using R_Arm_UR5
    13: Grab ISO-4762-M6-x-20 and screw into Spring-Hub using R_Arm_UR5
        -> Grab ISO-4762-M6-x-20 using R_Arm_UR5
        -> Screw ISO-4762-M6-x-20 to Spring-Hub using R_Arm_UR5
    14: Grab ISO-4762-M6-x-20 and screw into Spring-Hub using R_Arm_UR5
        -> Grab ISO-4762-M6-x-20 using R_Arm_UR5
        -> Screw ISO-4762-M6-x-20 to Spring-Hub using R_Arm_UR5
    15: Grab ISO-4034-M6 and screw into ISO-4762-M6-x-20 using R_Arm_UR5
        -> Grab ISO-4034-M6 using R_Arm_UR5
        -> Screw ISO-4034-M6 to ISO-4762-M6-x-20 using R_Arm_UR5
    16: Grab ISO-4034-M6 and screw into ISO-4762-M6-x-20 using R_Arm_UR5
        -> Grab ISO-4034-M6 using R_Arm_UR5
        -> Screw ISO-4034-M6 to ISO-4762-M6-x-20 using R_Arm_UR5
    17: Grab ISO-4034-M6 and screw into ISO-4762-M6-x-20 using R_Arm_UR5
        -> Grab ISO-4034-M6 using R_Arm_UR5
        -> Screw ISO-4034-M6 to ISO-4762-M6-x-20 using R_Arm_UR5
    18: Grab ISO-4034-M6 and screw into ISO-4762-M6-x-20 using R_Arm_UR5
        -> Grab ISO-4034-M6 using R_Arm_UR5
        -> Screw ISO-4034-M6 to ISO-4762-M6-x-20 using R_Arm_UR5
        -> Let go off Plate-Washer with L_Arm_UR5
```

Appendix B

AUTODESK® INVENTOR® 2020 API Object Model

December 20, 2018

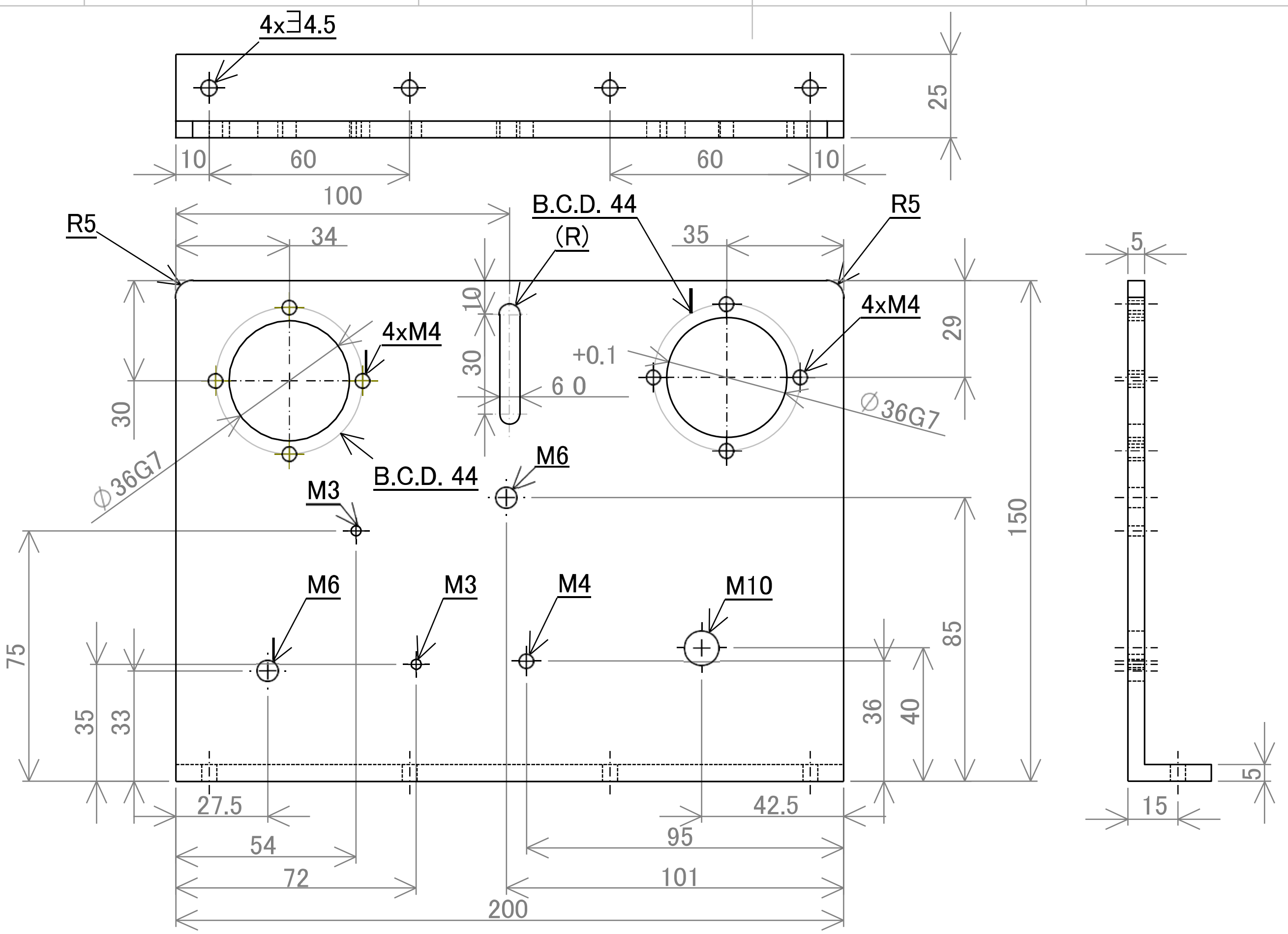

Application
Leaders
Point Clouds
Attributes
Model Annotations
Apprentice
2D Sketch
3D Sketch
Parameters
Transient Geometry & Objects
Geometry
Math Objects
Utility Objects
Client Graphics
B-Rep
Assets
(Materials and Appearances)
Dynamic Simulation
BIM Exchange
Legend
Collection Objects: Objects
Standard Objects: Object
New Objects in this release: Object
User Interface Related Objects
Part Document Related Objects
Assembly Document Related Objects
Drawing Document Related Objects
Presentation Document Related Objects
Base Class Objects: [x]
Derived Objects: (x)
See related object model diagram for this object at the bottom of the poster: *



* A5052 (No surface treatment) is used.

* Unless otherwise specified,
dimensional tolerances are according to
JIS B0405 Medium grade.

Part No.	Part name	Last update 2019-Oct.-24	
		Scale	Proj. method
000	01-base-01	0.500	3rd. angle projection
Designed by M. Shibata		World Robot Challenge 2020	
Checked by H. Dobashi			

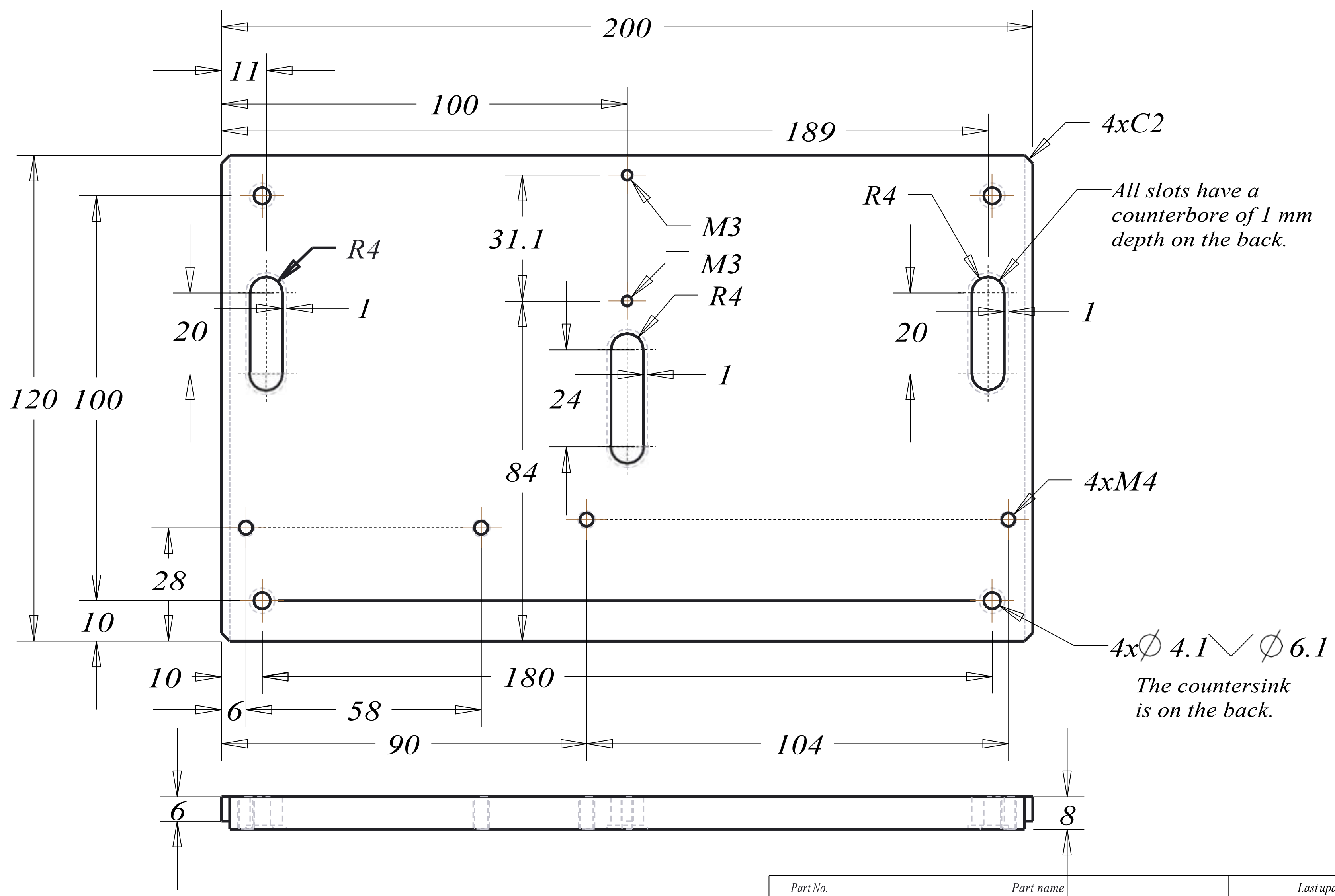


* A5052 (No surface treatment) is used.

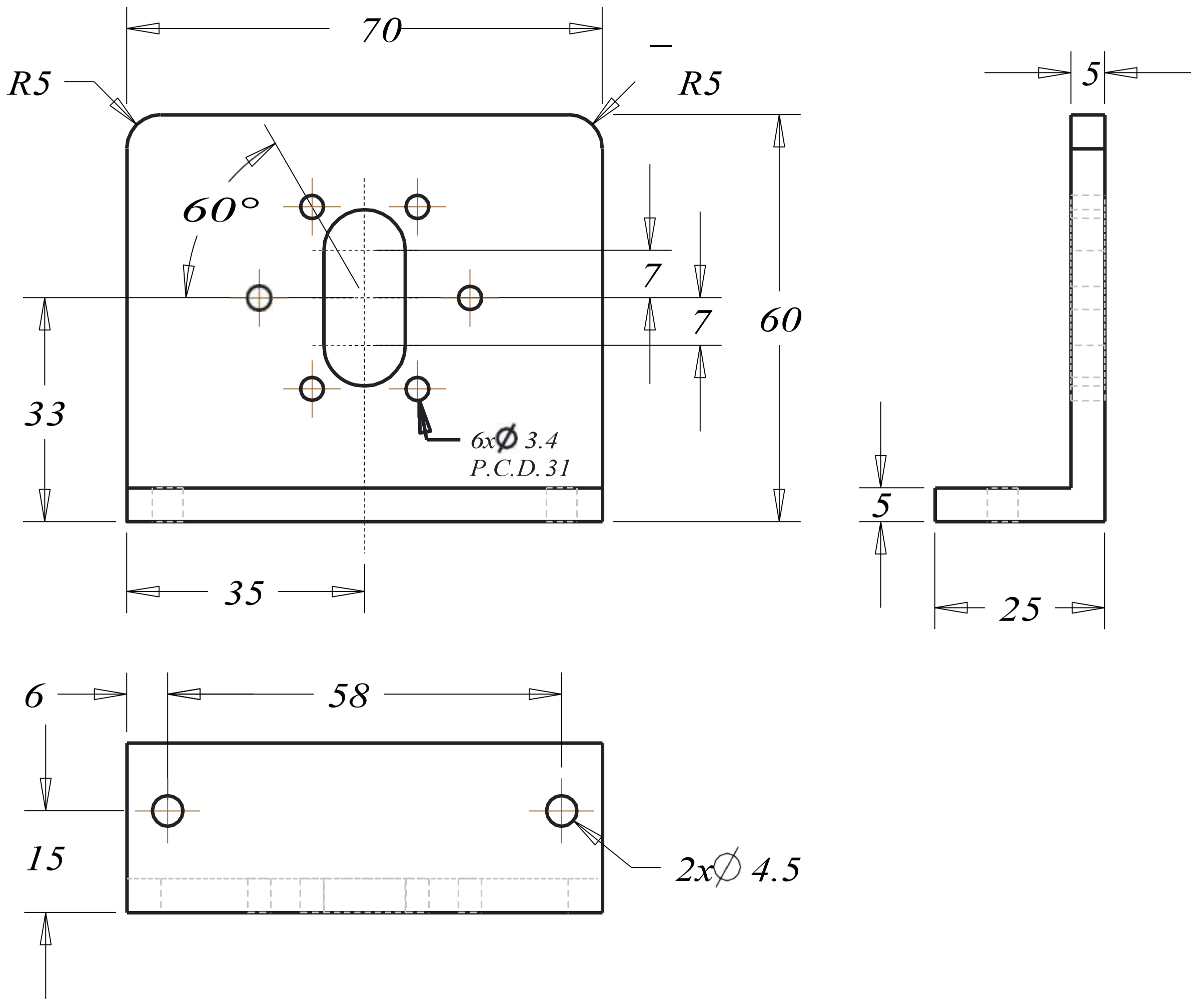
* Unless otherwise specified,
  dimensional tolerances are according to
  JIS B0405 Medium grade.

Part No.	Part name	Last update 2019-Oct.-24	
		Scale	Proj. method
000	02-plate-01	0.500	3rd. angle projection
Designed by M. Shibata			
Checked by H. Dobashi			

World Robot Challenge 2020


200
11
100
189
4xC2
R4
All slots have a counterbore of 1 mm depth on the back.
M3
M3
R4
31.1
R4
1
20
1
24
1
20
120
100
84
4xM4
28
10
10
180
6
58
90
104
4x∅ 4.1 ∨ ∅ 6.1
The countersink is on the back.
6
8

Part No.	Part name	Last update Jan-13-20	
		Scale	Proj. method
01	01-BASE	0.750	3rd. angle projection
Designer S. Kotosaka		World Robot Summit 2020 Assembly Task	
01-BASE			

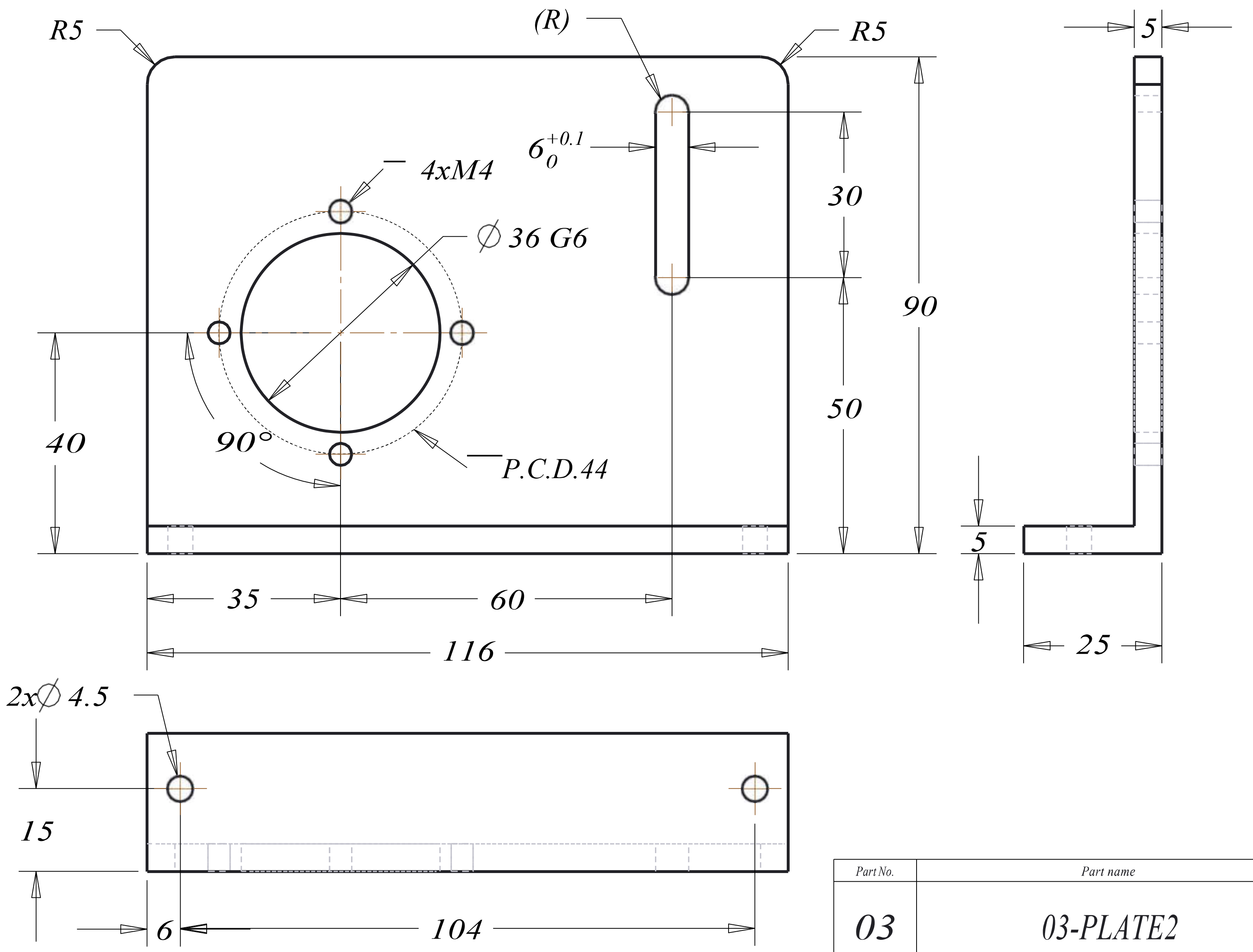


Part No.	Part name	Last update Jan-13-20	
		Scale	Proj. method
02	02-PLATE	1.000	3rd. angle projection

Designer S. Kotosaka

02-PLATE

World Robot Summit 2020 Assembly Task



Part No.	Part name	Last update Jan-13-20	
		Scale	Proj. method
03	03-PLATE2	1.000	3rd. angle projection
Designer S. Kotosaka		World Robot Summit 2020 Assembly Task	
03-PLATE2			

Assembly Task

Parts list for WRS2020 Assembly Task (without suprize task) 2020.Jan.14

No.	Name of part	Note.	Qty.	Retailer	MISUMI order number	Remarks
1	01–BASE	Base plate	1	manufactured		
2	02–PLATE	Output shaft fixing plate	1	manufactured		
3	03–PLATE2	Motor fixing plate	1	manufactured		
4	04_37D–GEARMOTOR–50–70	70:1 Metal Gearmotor 37Dx54L mm 12V (Helical Pinion)	1	Pololu	https://www.pololu.com/product/4744	with power cable No. 22, No. 23.
5	05_MBRFA30–2–P6	Pulley for Round Belt (4mm) – Setscrew P.D. 30mm	1	MISUMI	MBRFA30–2–P6	
6	06_MBT4–400	Polyurethane round belt (welded joint product) P.D. 4mm L=400mm	1	MISUMI	MBT4-400	
7	07_SBARB6200ZZ_30	Bearings with Housings (Double Bearings)	1	MISUMI	SBARB6200ZZ–30	
8	08_SSFHRT10–75–M4–FC55–G20	Drive shaft (Straight) D10h7	1	MISUMI	SSFHRT10–75–M4–FC55–G20	
9	09_EDCS10	End Cap for Shaft	1	MISUMI	EDCS10	
10	10_CLBPS10_17_4	Bearing Spacers For Inner Ring (output pulley)	1	MISUMI	CLBPS10–17–4	
11	11_MBRAC60–2–10	Pulley for Round Belts Clamping Type P.D. 60mm	1	MISUMI	MBRAC60–2–10	
12	12_CLBUS6–9–9.5	Bearing Spacers For Inner Ring (tension pulley)	1	MISUMI	CLBUS6–9–9.5	
13	13_MBGA30–2	Idler for Round Belt – Wide	1	MISUMI	MBGA30–2	
14	14_BGPSL6–9–L30–F7	Bearing Shaft Screw	1	MISUMI	BGPSL6–9–L30–F7	
15	15_SLBNR6	M6 Hex Nut (Fixing for idler shaft)	1	MISUMI	SLBNR6	
16	16_SPWF6	M6 Flat Washer (Fixing for idler shaft)	2	MISUMI	SPWF6	
17	17_SCB4–10	10mm M4 Socket Head Cap Screw (metric coarse thread)	9	MISUMI	SCB4–10	
18	18_SCB3–10	10mm M3 Socket Head Cap Screw (metric coarse thread)	6	MISUMI	SCB3–10	
19	19_MSSFS3–6	6mm M3 Hex Socket Set Screw (metric coarse thread)	1	MISUMI	MSSFS3–6	
20	20_TW1004	Clutch lock terminal block (compact)	1	MISUMI	TW1004	Supplied with part number 1 (base panel) attached with screw.
21	21_H0.5/14D	Weidmuller Wire–end ferrule	2	MISUMI	H0.5/14D	Crimped on the ends of the wires of part numbers 22 and 23.
22	22_NAUL1015_22_BK	NAUL1015 UL compliant wire (black)	1	MISUMI	NAUL1015–22–BK–10	Motor power line. Supplied with soldered to motor.
23	23_NAUL1015_22_R	NAUL1015 UL compliant wire (red)	1	MISUMI	NAUL1015–22–R–10	Motor power line. Supplied with soldered to motor.

Bibliography

Alfadhlani, T. M. A. Ari Samadhi, Anas Ma'ruf, & I. Setiasyahs.A. Toha (2011) “Automatic Collision Detection For Assembly Sequence Planning Using A Three-Dimensional Solid Model.” 10 Journal Of Advanced Manufacturing Systems. 277-291.

Almassri, Ahmed M., W. Z. Wan Hasan, Siti Anom Ahmad, Asnor J. Ishak, am Ghazali, D. N. Talib, & Chikamune Wada (2015) “Pressure sensor: state of the art, design, and application for robotic hand.” 2015 Journal of Sensors.

Autodesk Autodesk Inventor https://www.autodesk.com/products/inventor/overview.

Autodesk Inventor “Autodesk Inventor Object Model,” https://knowledge.autodesk.com/akn-aknsite-article-attachments/d20aa033-13a7-4b23-a790-1897b317c523.pdf.

Autodesk Inventor “Inventor API User's Manual,” https://help.autodesk.com/view/INVNTOR/2018/ENU/?guid=GUID-5901102A-F148-4CD4-AF50-26E2AFDEE6A7.

Bjorkelund, Anders, Lisett Edstrom, Mathias Haage, Jacek Malec, Klas Nilsson, Pierre Nugues, Sven Gestegard Robertz, Denis Storkle, Anders Blomdell, Rolf Johansson, Magnus Linderoth, Anders Nilsson, Anders Robertsson, Andreas Stolt, & Herman Bruyninckx, eds. (2011) On the integration of skilled robot motions for productivity in manufacturing, IEEE.

Bourjault, A. (1984) “Contribution to a methodological approach of automated assembly: automatic generation of assembly sequence.” University de Franche-Comte.

Bourjault, A. (1984) “Contribution to a Methodological Approach to Automated Assembly: Automatic Elaboration of Operational Sequences.” These D'etat, University of Franche-Comte.

Cappelleri, David J., & Zhenbo Fu (052013) “Towards flexible, automated microassembly with caging micromanipulation,” in, 2013 IEEE International Conference on Robotics and Automation, IEEE, p. 1427-32.

Chandra, Akkaladevi Sharath, Plasch Matthias, Pichler Andreas, & Rinner Bernhard (2016) “Human Robot Collaboration to Reach a Common Goal in an Assembly Process.” 284 Frontiers in Artificial Intelligence and Applications. 3-14.

Chaudron, V., P. Martin, & X. Godot (2005) “Assembly sequences: planning and simulating assembly operations,” in, (ISATP 2005). The 6th IEEE International Symposium on Assembly and Task Planning: From Nano to Macro Assembly and Manufacturing, 2005, IEEE, p. 156-61.

Colledanchise, Michele, & Petter Ögren (2018) Behavior trees in robotics and AI: An introduction, CRC Press.

Corporation, P. T. Creo Parametric https://www.ptc.com/en/products/creo/parametric.

Dobra, Andreea (92014) “General classification of robots. Size criteria,” in, 2014 23rd International Conference on Robotics in Alpe-Adria-Danube Region (RAAD), IEEE, p. 1-6.

Drigalski, Felix von, Chisato Nakashima, Yoshiya Shibata, Yoshinori Konishi, Joshua C. Triyonoputro, Kaidi Nie, Damien Petit, Toshio Ueshiba, Ryuichi Takase, Yukiyasu Domae, Taku Yoshioka, Yoshihisa Ijiri, Ixchel G. Ramirez-Alpizar, Weiwei Wan, & Kensuke Harada (2020) “Team O2AS at the world robot summit 2018: an approach to robotic kitting and assembly tasks using general purpose grippers and tools.” Advanced Robotics. 1-17.

Drigalski, Felix von, Christian Schlette, Martin Rudorfer, Nikolaus Correll, Joshua C. Triyonoputro, Weiwei Wan, Tokuo Tsuji, & Tetsuyou Watanabe (2019) "Robots assembling machines: learning from the World Robot Summit 2018 Assembly Challenge." Advanced Robotics. 1-14.

Ebinger, Timothy, Sascha Kaden, Shawna Thomas, Robert Andre, Nancy M. Amato, & Ulrike Thomas (52018) "A General and Flexible Search Framework for Disassembly Planning," in, 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, p. 3548-55.

Eda Kavlakoglu (2020) "AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the Difference?," https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks.

Eng, Tiam-Hock, Zhi-Kui Ling, Walter Olson, & Chuck McLean (1999) "Feature-based assembly modeling and sequence generation." 36 Computers & Industrial Engineering. 17-33.

Fazio, T. de, & D. Whitney (1987) "Simplified generation of all mechanical assembly sequences." 3 IEEE Journal on Robotics and Automation. 640-658.

Festo (2020) "White paper: Robots," https://www.festo.com/media/cms/media/editorial/WP_Robots_en_V01_2019.pdf (accessed 27 September 2020).

Fujita, M., Y. Domae, A. Noda, G. A. Garcia Ricardez, T. Nagatani, A. Zeng, S. Song, A. Rodriguez, A. Causo, I. M. Chen, & T. Ogasawara (2019) "What are the important technologies for bin picking? Technology analysis of robots in competitions based on a set of performance metrics." Advanced Robotics. 1-15.

Garcıa, Luis Enrique Ramos (2010) "Ontological CAD data interoperability framework.".

Gilday, Kieran, Josie Hughes, & Fumiya Iida (102018) "Achieving Flexible Assembly Using Autonomous Robotic Systems," in, 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, p. 1-9.

Inc, S. P. L. M. S. Solid Edge https://solidedge.siemens.com/en/.

Inc., S. P. L. M. S. NX https://www.plm.automation.siemens.com/global/en/products/nx/nx-for-design.html.

International Federation of Robotics "Industrial Robots," https://ifr.org/industrial-robots (accessed 27 September 2020).

ISO 8373 (2012) "Robots and robotic devices — Vocabulary," https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-2:v1:en.

Keijsers, N.L.W. (2010) "Neural Networks," in, Encyclopedia of Movement Disorders, Elsevier, p. 257-59.

Kojima, T., Y. Kikuchi, S. Seki, & H. Wakiwaka (2004) "Study on high accuracy optical encoder with 30 bits," in, The 8th IEEE International Workshop on Advanced Motion Control, 2004. AMC '04, IEEE, p. 493-98.

Koubaa, Anis (2017) Robot Operating System (ROS), Cham: Springer International Publishing.

Lit, Pierre de, Patrice Latinne, Brahim Rekiek, & Alain Delchambre (2001) "Assembly planning with an ordering genetic algorithm." 39 International Journal of Production Research. 3623-3640.

Liu, G. (2002) "On velocity estimation using position measurements," in, Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301), IEEE, 1115-1120 vol.2.

Malik, Ali Ahmad, & Arne Bilberg (2018) "Digital twins of human robot collaboration in a production setting." 17 Procedia Manufacturing. 278-285.

Markus Ikeda, Srinivas Maddukuri, Michael Hofmann, Andreas Pichler, Xiang Zhang,Athanasios Polydoros, Justus Piater, Klemens Winkler, Klaus Brenner,Ioan Harton and Uwe Neugebauer in Markus Ikeda, Srinivas Maddukuri, Michael Hofmann, Andreas Pichler, Xiang Zhang,Athanasios Polydoros, Justus Piater, Klemens Winkler, Klaus Brenner,Ioan Harton and Uwe Neugebauer, ed., Proceedings of the Austrian Robotics Workshop 2018, innsbruck university press.

Markus Ikeda, Srinivas Maddukuri, Michael Hofmann, Andreas Pichler, Xiang Zhang,Athanasios Polydoros, Justus Piater, Klemens Winkler, Klaus Brenner,Ioan Harton and Uwe Neugebauer, ed. Proceedings of the Austrian Robotics Workshop 2018, innsbruck university press.

Mathew, Arun Tom, & C. S. P. Rao (2010) "A Novel Method of Using API to Generate Liaison Relationships from an Assembly." 03 Journal of Software Engineering and Applications. 167-175.

Matt Minner (2019) "4 Types of Robots Every Manufacturer Should Know," https://www.nist.gov/blogs/manufacturing-innovation-blog/4-types-robots-every-manufacturer-should-know (accessed 27 September 2020).

Misumi "Misumi," https://us.misumi-ec.com/.

Mitsi, S., K.-D. Bouzakis, G. Mansour, D. Sagris, & G. Maliaris (2005) "Off-line programming of an industrial robot for manufacturing." 26 The International Journal of Advanced Manufacturing Technology. 262-267.

Nie, Kaidi, Felix von Drigalski, Joshua C. Triyonoputro, Chisato Nakashima, Yoshiya Shibata, Yoshinori Konishi, Yoshihisa Ijiri, Taku Yoshioka, Yukiyasu Domae, Toshio Ueshiba, Ryuichi Takase, Xinyi Zhang, Damien Petit, Ixchel G. Ramirez-Alpizar, Weiwei Wan, & Kensuke Harada (2020) "Team O2AS' approach for the task-board task of the World Robot Challenge 2018." Advanced Robotics. 1-22.

Nilsson, Klas (1996) Industrial robot programming, Department of Automatic Control, Lund Institute of Technology Lund.

Ou, Li-Ming, & Xun Xu (2013) "Relationship matrix based automatic assembly sequence generation from a CAD model." 45 Computer-Aided Design. 1053-1067.

Perzylo, Alexander, Nikhil Somani, Markus Rickert, & Alois Knoll (92015) "An ontology for CAD data and geometric constraints as a link between product models and semantic robot task descriptions," in, 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, p. 4197-203.

Polydoros, Athanasios S., Bjarne Großmann, Francesco Rovida, Lazaros Nalpantidis, & Volker Krüger (2016) "Accurate and Versatile Automation of Industrial Kitting Operations with SkiROS," in, Towards Autonomous Robotic Systems, Springer International Publishing, 255–268-255–268.

Poole, Harry H. (2012) Fundamentals of robotics engineering, Springer Science & Business Media.

Rovida, Francesco, Matthew Crosby, Dirk Holz, Athanasios S. Polydoros, Bjarne Großmann, Ronald P. A. Petrick, & Volker Krüger (2017) "SkiROS—A Skill-Based Robot Control Platform on Top of ROS," in, Studies in Computational Intelligence, Springer International Publishing, 121–160-121–160.

Sadrfaridpour, Behzad, Hamed Saeidi, & Yue Wang (82016) "An integrated framework for human-robot collaborative assembly in hybrid manufacturing cells," in, 2016 IEEE International Conference on Automation Science and Engineering (CASE), IEEE, p. 462-67.

Saitou, Kazuhiro, ed. (82015 // 2015?) 2015 IEEE International Conference on Automation Science and Engineering (CASE): Automation for a Sustainable Future : Gothenburg, Sweden, August 24-28, 2015, [Piscataway, New Jersey]: IEEE.

Schaal, Stefan (1999) "Is imitation learning the route to humanoid robots?" 3 Trends in Cognitive Sciences. 233–242-233–242.

Schlette, C., A. G. Buch, F. Hagelskjær, I. Iturrate, D. Kraft, A. Kramberger, A. P. Lindvig, S. Mathiesen, H. G. Petersen, M. H. Rasmussen, T. R. Savarimuthu, C. Sloth, L. C. Sørensen, & T. N. Thulesen (2019) "Towards robot cell matrices for agile production – SDU Robotics' assembly cell at the WRC 2018." Advanced Robotics. 1-17.

Shibata, Mizuho, Hiroki Dobashi, Wataru Uemura, Shinya Kotosaka, Yasumichi Aiyama, Takeshi Sakaguchi, Yoshihiro Kawai, Akio Noda, Kazuhito Yokoi, & Yasuyoshi Yokokohji (2020) "Task-board task for assembling a belt drive unit." Advanced Robotics. 1–23-1–23.

Soetebier, Sven, Christian Muller, Nicolas Mauser, Sonke Kock, & Fabrice Legeleux (082008) "Flexible automation for automotive body assembly," in, 2008 IEEE International Conference on Automation Science and Engineering // IEEE-CASE 2008, [Piscataway, NJ]: IEEE, p. 341-46.

Su, Qiang (2007) "Computer aided geometric feasible assembly sequence planning and optimizing." 33 The International Journal of Advanced Manufacturing Technology. 48-57.

Systèmes, D. CATiA http://www.3ds.com/products/catia.

Systèmes, D. D. Systèmes. SolidWorks https://www.solidworks.com/product/solidworks-3d-cad.

Tajima, Sho, Seiji Wakamatsu, Taiki Abe, Masanari Tennomi, Koki Morita, Hirotoshi Ubata, Atsushi Okamura, Yuji Hirai, Kota Morino, Yosuke Suzuki, Tokuo Tsuji, Kimitoshi Yamazaki, & Tetsuyou Watanabe (2019) "Robust bin-picking system using tactile sensor." Advanced Robotics. 1-15.

Tennomi, Masanari, Atsushi Okamura, Yuta Nakamura, Taiki Abe, Seiji Wakamatsu, Sho Tajima, Toshihiro Nishimura, Yuji Hirai, Takuro Sawada, Naoki Ichikawa, Tokuo Tsuji, Kimitoshi Yamazaki, Yosuke Suzuki, & Tetsuyou Watanabe (2020) "Development of assembly system with quick and low-cost installation." Advanced Robotics. 1-15.

Thomas, Ulrike, Theodoros Stouraitis, & Maximo A. Roa (82015 // 2015?) "Flexible assembly through integrated assembly sequence planning and grasp planning," in K. Saitou, ed., 2015 IEEE International Conference on Automation Science and Engineering (CASE): Automation for a Sustainable Future : Gothenburg, Sweden, August 24-28, 2015, [Piscataway, New Jersey]: IEEE, p. 586-92.

Udacity (2020) "Robotics Software Engineer," https://www.udacity.com/course/robotics-software-engineer--nd209.

Wang, Yan, Kensuke Harada, & Weiwei Wan (2020) "Motion planning of skillful motions in assembly process through human demonstration." 34 Advanced Robotics. 1079-1093.

Weik, Martin H., ed. (2001) Computer Science and Communications Dictionary, Boston, MA: Springer US.

Weik, Martin H. (2001) "robot," in M. H. Weik, ed., Computer Science and Communications Dictionary, Boston, MA: Springer US, p. 1499.

World Robot Summit "World Robot Summit Assembly Task Rulebook," https://worldrobotsummit.org/wrs2020/challenge/download/Rules/DetailedRules_Assembly_EN.pdf.

WRS (2020) Industrial Robotics Category Assembly Challenge Rule Book, WRS.

Yokokohji, Yasuyoshi, Yoshihiro Kawai, Mizuho Shibata, Yasumichi Aiyama, Shinya Kotosaka, Wataru Uemura, Akio Noda, Hiroki Dobashi, Takeshi Sakaguchi, & Kazuhito Yokoi (2019) "Assembly Challenge: a robot competition of the Industrial Robotics Category, World Robot Summit – summary of the pre-competition in 2018." 33 Advanced Robotics. 876-899.

Yoshikawa, Tsuneo (1990) Foundations of robotics: analysis and control, MIT press.

Zha, X.F, & H. Du (2002) "A PDES/STEP-based model and system for concurrent integrated design and assembly planning." 34 Computer-Aided Design. 1087-1110.