Automatic Grouping of Link Components for Kinematic Chains of Multi-Axis Machine Tools

by

Minsi Sung

B.Eng, National Cheng Kung University, 2018

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Mechanical Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

January 2021

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Automatic Grouping of Link Components for Kinematic Chains of Multi-Axis Machine Tools

submitted by Minsi Sung in partial fulfillment of the requirements for

the degree of Master of Applied Science

in Mechanical Engineering

Examining Committee:

Dr. Hsi-Yung (Steve) Feng, Professor, Department of Mechanical Engineering, UBC
Supervisor

Dr. Yusuf Altintas, Professor, Department of Mechanical Engineering, UBC
Supervisory Committee Member

Dr. Xiaoliang Jin, Assistant Professor, Department of Mechanical Engineering, UBC
Supervisory Committee Member

Abstract

Multi-axis machine tools are used to machine parts with complex, curved surfaces. With additional rotary axes, multi-axis machine tools have a higher risk to have collisions compared with traditional three-axis machine tools. Collision during machining often causes damages to the machine and workpiece, which in turn leads to loss of productivity and extra costs. It can occur between the cutter, workpiece, and machine. Machining simulation with moving machine axis links becomes essential to detecting collisions prior to physical machining. In order to simulate machine movements, it is necessary to attain the kinematic chain of a given machine tool and to group machine components for each link in the kinematic chain. Existing methods to group link components require many inputs from users and follow an error-prone and lengthy manual process.

This thesis presents an automatic method to group link components for each machine axis of a given multi-axis machine tool. The method is able to generate the kinematic chain of the multi-axis machine tool with only basic user inputs. As the first step, interference detection by voxel modeling is used to get contact relationships between components. Link-interface features between components are then identified and used to generate the link groups. The process of generating the link groups may be accompanied with uncertainties that can result in incorrect link groups. As a result, if there is an uncertainty, the generated link groups need to be validated to be free of link collision within the travel span of each axis. If there is collision, the collision is to be resolved by examining the uncertainty causing the specific link collision. The iterative step of validation and resolution continues until no link collision exists. The link collision is detected also by voxel modeling. The output of the automatic grouping method is the kinematic chain of the machine tool

and the geometric model of each link for machining simulation. The presented method has been implemented on five commercial Haas five-axis machine tools with varying configurations. Correct kinematic chains for these machine tools have been generated and ready to be used for simulation of machine movements.

Lay Summary

This thesiss presents an automatic grouping method that generates kinematic chains and corresponding STL files of each link of multi-axis machine tools for simulation of machine movements. It requires only basic user inputs to avoid an error-prone and lengthy manual process. If necessary, validation of link groups and resolution of link collision are conducted to ensure correct link groups are generated. The presented method has been implemented on five commercial Haas five-axis machine tools with varying configurations. Correct kinematic chains for these machine tools have been generated and ready to be used for simulation of machine movements.

Preface

This thesis is original, unpublished, independent work by the author, Minsi Sung. Dr. Hsi-Yung (Steve) Feng is the supervisor for this work.

Table of Contents

Abstract ..... **iii**

Lay Summary ..... **v**

Preface ..... **vi**

Table of Contents ..... **vii**

List of Tables ..... **ix**

List of Figures ..... **x**

List of Abbreviations ..... **xii**

Acknowledgements ..... **xiii**

Chapter 1: Introduction **1**

1.1 Motivation 5

1.2 Objectives 6

1.3 Organization 6

Chapter 2: Literature Review **7**

2.1 Multi-Axis Machine Tool Configurations 7

2.2 Kinematic Chains for Motion Simulation 11

2.3 Interference Detection and Voxel Modelling 14

2.4 Generation of Kinematic Chains of Multi-Axis Machine tools 16

Chapter 3: Methodology **18**

3.1 Required Inputs 18

3.2 Automatic Grouping Method 21

Chapter 4: Link-Component Grouping **25**

4.1 Contact-Components Pairs 25

4.2 Link-Interface Pairs 27

4.3 Generation of Link Groups 39

Chapter 5: Validation of Link Groups and Resolution of Link Collision 45

5.1 Validation of Link Groups 45

5.2 Resolution of Link Collision 53

Chapter 6: Implementation Results 56

6.1 Single-Level Voxel Size 57

6.2 Two-Level Voxel Size 69

Chapter 7: Conclusions and Future Works 72

7.1 Conclusions 72

7.2 Future Works 73

Bibliography 74

List of Tables

Table 4-1 Criteria of translational LIPs 36

Table 4-2 Criteria of rotary LIPs 37

Table 5-1 Summary of required voxelization and parent voxel model number for five-axis machine tools 52

Table 6-1 Results of VF-2TR using strategy A (Single-level voxel size) 58

Table 6-2 Results of VF-2TR using strategy B (Single-level voxel size) 58

Table 6-3 Results of UMC-500 using strategy A (Single-level voxel size) 59

Table 6-4 Results of UMC-500 using strategy B (Single-level voxel size) 59

Table 6-5 Results of UMC-750 using strategy A (Single-level voxel size) 60

Table 6-6 Results of UMC-750 using strategy B (Single-level voxel size) 60

Table 6-7 Results of UMC-1600H using strategy A (Single-level voxel size) 61

Table 6-8 Results of UMC-1600H using strategy B (Single-level voxel size)..... 61

Table 6-9 Results of VR-8 using strategy A (Single-level voxel size)..... 62

Table 6-10 Results of VR-8 using strategy B (Single-level voxel size)..... 62

Table 6-11 Results of VF-2TR using strategy A (two-level voxel size) 69

Table 6-12 Results of VF-2TR using strategy B (two-level voxel size)..... 70

Table 6-13 Results of UMC-500 using strategy A (two-level voxel size) 70

Table 6-14 Results of UMC-500 using strategy B (two-level voxel size)..... 71

List of Figures

Figure 1.1 Example of five-axis machine tool 2

Figure 1.2 Collision between machine head and clamping table [6] 2

Figure 1.3 Link components of a link 4

Figure 2.1 Tree-shaped chart of a machine tool [13] 8

Figure 2.2 Three structural types of five-axis machine tools (a) Rotary table(b) Spindle rotating (c) Hybrid [16] 10

Figure 2.3 Visualization of URDF's basic language elements: links and joints [22] 12

Figure 2.4 URDF example 12

Figure 2.5 Voxelization Steps [31] 15

Figure 2.6 Configuration selection dialog [39] 17

Figure 2.7 Assembly dialog [39] 17

Figure 3.1 Flowchart of inputs and outputs 18

Figure 3.2 Automatic grouping method flowchart 22

Figure 4.1 CCP map of UMC-750 26

Figure 4.2 False interference in 2D 27

Figure 4.3 CCP map with LIPs of UMC-750 28

Figure 4.4 Original (left) and offset triangular mesh model of C-axis of UMC-750 (right) 31

Figure 4.5 Original (left) and offset triangular mesh model of B-axis of UMC-750 (right) 31

Figure 4.6 Original (left) and offset triangular mesh model of machine base of UMC-750 (right) 31

Figure 4.7 Original (Left) and offset voxel model of machine of C-axis of UMC-750 (right) 32

Figure 4.8 Original (Left) and offset voxel model of machine of B-axis of UMC-750 (right) 32

Figure 4.9 Original (Left) and offset voxel model of machine base of UMC-750 (right)..... 32

Figure 4.10 Two mesh models of components of a X-LIP..... 38

Figure 4.11 Two offset voxel models of components of a X-LIP (a) before (b) after translating 39

Figure 4.12 CCP map of UMC-500 example 40

Figure 4.13 Link-component grouping of UMC-500 example after step 1..... 41

Figure 4.14 Link-component grouping of UMC-500 example after step 2..... 43

Figure 4.15 Final result of link-component grouping of UMC-500 example..... 44

Figure 5.1 CCP map of UMC500-CCPs using 1.3mm voxel size..... 46

Figure 5.2 Example of configuration types of five-axis machine tool (a) 5-0 (b) 4-1 (c) 3-2..... 50

Figure 5.3 Link-component grouping of UMC-500 example after step 1..... 54

Figure 5.4 Final result of link-component grouping of UMC-500 example..... 55

Figure 6.1 Kinematic chain demonstration of VF-2 63

Figure 6.2 Kinematic chain demonstration of UMC-500 64

Figure 6.3 Kinematic chain demonstration of UMC-1600H 65

Figure 6.4 Kinematic chain demonstration of VR-8..... 66

List of Abbreviations

B-rep - boundary representation

CAD - Computer-aided design

CCP - Contact-component pair

IGES - Global Environmental Strategies

LIP - Link-interface pair

ROS - Robot Operating System

URDF - Unified Robotic Description Format

Acknowledgements

Many thanks to Dr. Hsi-Yung (Steve) Feng for his guidance and support. My sincere gratitude to my family and friends for their support and encouragement. Thank you to the National Science and Engineering Research Council of Canada (NSERC) for their funding of this research.

Chapter 1: Introduction

Multi-axis (four-axis or five-axis) machine tools are widely used to machine parts with complex and curved surfaces. The number of axes of a machine tool refers to the number of degrees of freedom. A three-axis milling machine has three linear translational axes X, Y, and Z which can be positioned everywhere within the travel span of each axis. To increase the flexibility in possible tool-workpiece orientations, it can be achieved by providing rotary axes. The rotary axes are defined as A, B, and C axis which rotates about the X, Y, and Z axis, respectively. Figure 1.1 shows an example of a five-axis machine tool with rotary axes in the B and C axes. Compare with three-axis machine tools, multi-axis machine tools offer advantages of better productivity, accuracy, and flexibility with additional rotary axes [1, 2].

However, because of additional rotary axes, collision is prone to occur during multi-axis machining. Collision can bring destructive consequences like damages to tools, workpieces, and even the machine structure. It results in less productivity and extra costs. The collision between the machine head and clamping table during the 5-axis machining of a larger propeller blade is shown in Figure 1.2. Collision greatly affects the wide application of multi-axis machines and has attracted many researchers in collision detection for multi-axis machining [3-5].

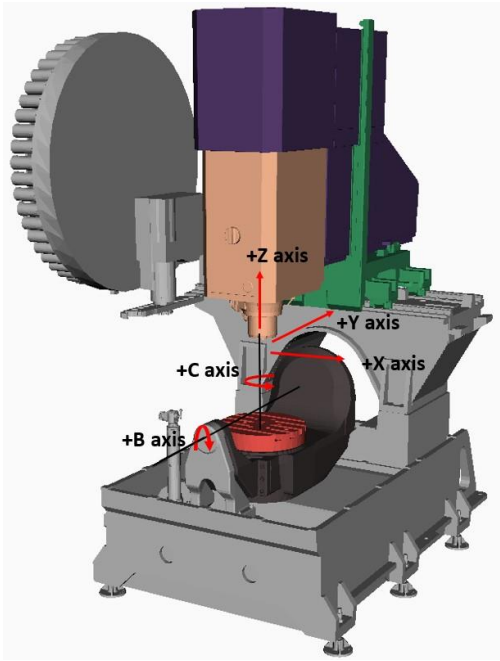

Figure 1.1 Example of five-axis machine tool

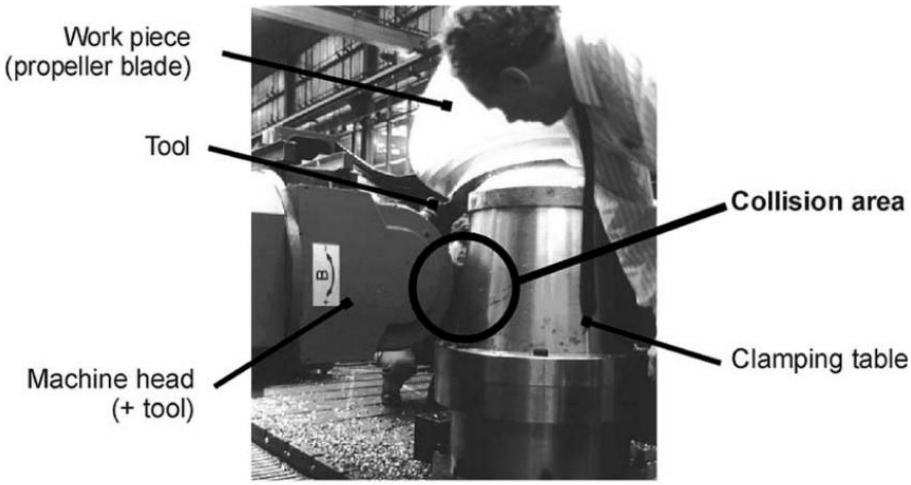

Figure 1.2 Collision between machine head and clamping table [6]

In order to avoid collisions of multi-axis machine tools, it is essential to have a virtual machine tool (machine tool motion simulation program) [7] to detect collisions before physical machining [6]. Constructing a virtual machine tool requires geometric models of movable axes and the machine base of the given machine tool. Also, the machine kinematic chain and machining commands are necessary. A kinematic chain includes the information of links and joints. A link describes a rigid body with its geometric shape. A joint describes how two links are connected and how they can move relative to each other.

Multi-axis machine tools are considered to be a special form of robots. Multi-axis machines have a limited number of configurations and robots have infinite configurations. For a five-axis machine tool, it contains six links representing five movable axes and the machine base. Also, it has five joints representing the five degrees of freedom between six links. For a four-axis machine tool, it consists of five links and four joints. There are many various pieces of software with good user-friendly interfaces that assist engineers in simulating different types of robots that have different kinematics chains. Simulating multi-axis machine movements and generating kinematic chains of machines can follow the same process as robots. Robot Operating System (ROS) [8] is a set of open-source software libraries and tools that help engineers build robot applications. ROS uses Unified Robotic Description Format (URDF) which is an XML file format to describe all elements of a robot for simulating movements of robots in a virtual environment. URDF not only describes the kinematic chain of a given robot but also includes the information of the corresponding geometric model for each link. The geometric model of each link can be formed by a group of components instead of solely one component. The components that constitute a link are called the link components of this link. A link group indicates a set of link components that are grouped

together to form a link and these link components always move together during simulating robot movements. For example, component 1 and component 2 in Figure 1.3 are the link components of this link. These two link components are grouped together to form a link group of this link.

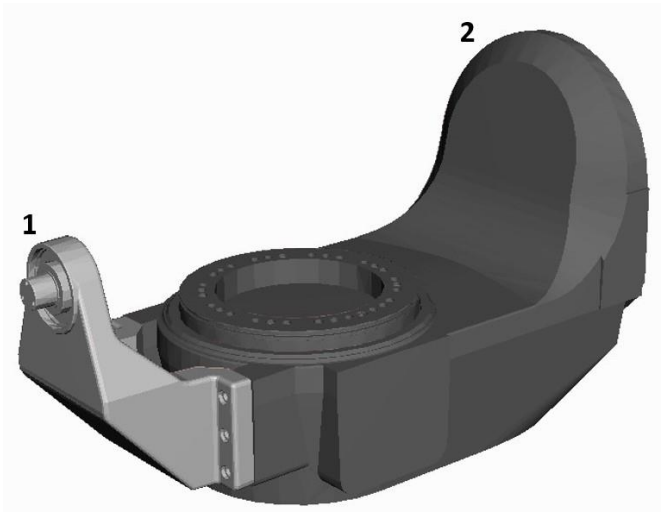

Figure 1.3 Link components of a link

1.1 Motivation

Multi-axis machine tools that have a limited number of configurations are considered to be a special form of robots, so the kinematic chains of machines can be described in URDF. Obtaining URDF of multi-axis machine tools requires many inputs to define the properties of machines, such as locations and orientations of reference rotary axes of rotary links, joint types, and the configuration tree of links, etc. Grouping link components for each link is also required.

Although assembly files of multi-axis machines have already included the pre-defined link groups, they are error-prone because the link components are grouped with different styles of different robot designers instead of following how link components should be grouped to simulate machine movements. Thus, it still requires grouping link components in order to generate URDF of machines for motion simulation. Manually grouping link components and defining the necessary properties to generate URDF of multi-axis machines can be done by URDF Exporter [9] that requires the assembly files of machines.

Manually specifying all required information of a given multi-axis machine tool for exporting its URDF is a lengthy process. Moreover, because the necessary manual inputs and grouping rely on high-required machine tool knowledge leading to an error-prone process by users. It is desirable to have an automatic grouping method to generate URDF for multi-axis machine tools with only basic manual inputs. Unfortunately, there is a lack of work on grouping link components automatically to generate kinematic chains of multi-axis machine tools for simulation of machine movements.

1.2 Objectives

Given the lack of work on grouping link components and generating kinematic chains for multi-axis machine tools automatically, there is evidently room for new research in this area. The feasible configurations of multi-axis machine tools and methods of generating kinematic chains for robots are studied in order to analyze the basic inputs required from users. In comparison to existing methods that require many user inputs with a lengthy manual process, this thesis presents an automatic method to group link components for each machine axis of a given multi-axis machine tool with contact information using voxel modeling. It can generate kinematic chains of multi-axis machine tools for simulation of machine movements with only basic inputs from users.

1.3 Organization

This thesis is organized as follows: Chapter 2 is the literature review of multi-axis machine tool configurations, kinematic chains for motion simulation, interference detection and voxel modeling, and then generation of kinematic chains of multi-axis machine tools. Chapter 3 describes the inputs and outputs of each tool used in the presented automatic grouping method. Also, the workflow of the method is discussed. Details of the link-component grouping, validation of link groups, and resolution of link collision are elaborated in Chapter 4 and Chapter 5 respectively. The presented method has been implemented on five commercial Haas [10] five-axis machine tools with varying configurations, and the results are shown in Chapter 6. Finally, Chapter 7 presents the conclusions of the work and suggests future work that could be done to improve it.

Chapter 2: Literature Review

This chapter presents and discusses the state of the art of multi-axis machine tool configurations, kinematic chains for motion simulation, and interference detection and voxel modeling. In addition, the state of the art of generation of kinematic chains of multi-axis machine tools is included for discussion.

2.1 Multi-Axis Machine Tool Configurations

The most common multi-axis machine tools are four-axis and five-axis machine tools. Machine tools that contain more than five axes are not common because five degrees of freedom are the minimum required to obtain maximum flexibility in tool-workpiece orientation. This means that the tool and workpiece can be oriented relative to each other under any angle [11].

In order to represent configurations of multi-axis machine tools, Sakamoto [12] presented the configuration code K to represent the sequence of the relative movements of links of a machine tool from the beginning of the working table to the spindle as:

$$\mathrm{K} = k_1 k_2, \cdots, 0, \cdots, k_{N-1} k_N$$

where $k_i =$ x, y or z which represents the relative translation of two machine links along the X, Y, and Z axis directions, or $k_i =$ a, b or c, representing the relative rotation of two machine links about the X, Y, and Z axes; subscript 0 refers to the machine base. Because a configuration code only specifies the order of axes of a machine tool, it lacks the capability to describe relative movements or positions between axes of the machine. Mei et al. [13] used a tree-shaped chart to

represent the configuration of a virtual machine tool and an example of tree-shaped chart of a machine tool is shown in Figure 2.1. Each node in the tree-shaped chart represents the coordinate system of a link, and each arrow between nodes is a coordinate transformation matrix. Each link coordinate system in a virtual machine tool can be derived by sequentially multiplying the coordinate transformation matrices from the machine base to that component. As a result, using a tree-shaped chart to represent a virtual machine tool can describe relative movements or positions between axes of the machine. Also, controlling the coordinate transformation matrices between links can drive the motion of the virtual machine tool.

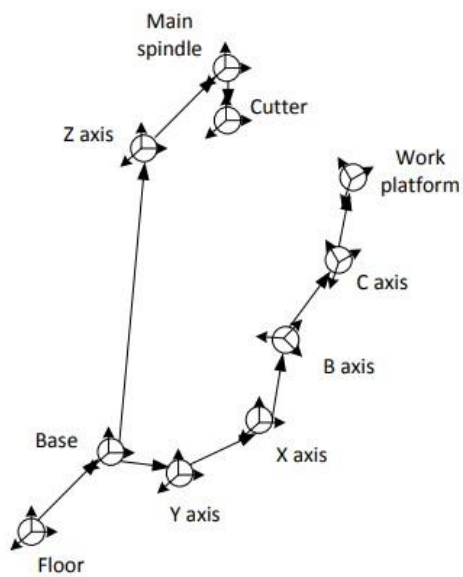

Figure 2.1 Tree-shaped chart of a machine tool [13]

Four-axis machine tools consist of three linear translational axes X, Y, and Z plus a machine base and a rotary axis, either A, B, or C axis. Therefore, the configuration code is a five-digit number

with $5!C_1^3$ or 360 permutations. For five-axis machines, there are two rotary axes which can be any two of axes A, B, and C. Its configuration code is formed by a six-digit number comprising $6!C_2^3 = 2160$ permutations. It is noted that not all the configurations are feasible for four-axis and five-axis machine tools because the independence of three linear translational axes is compulsory. Hence, two configuration limitations on four-axis and five-axis machine tools are identified:

1. For five-axis machine tools, when the fourth and fifth axes are A (or B) axis, and C axis respectively, C axis must be the one nearer to the working table. As for four-axis machine tools, the fourth axis cannot be C axis.
2. There must not be a rotary axis between any two of the three linear translational axes and one linear translational axis must be attached to the machine base.

Because of these limitations, there are 96 feasible configurations of four-axis machine tools and 288 feasible configurations of five-axis machine tools [14].

There was no standard terminology available for the feasible configurations of five-axis machine tools and almost every relevant paper has been using different terms to refer to the same basic machine types [15-17]. Therefore, Tutunea-Fatan and Feng [18] grouped feasible five-axis machine tool configurations into three structural types in order to avoid confusion:

1. Rotary table (RT): the two rotational movements are applied to the workpiece
2. Spindle rotating (SR): the two rotational movements are applied to the cutting tool
3. Hybrid (HT): one rotation is applied to the cutting tool and the other to the workpiece

This naming convention of feasible five-axis machine tool configurations are followed in this thesis. Figure 2.2 illustrates three structural types of five-axis machine tools.

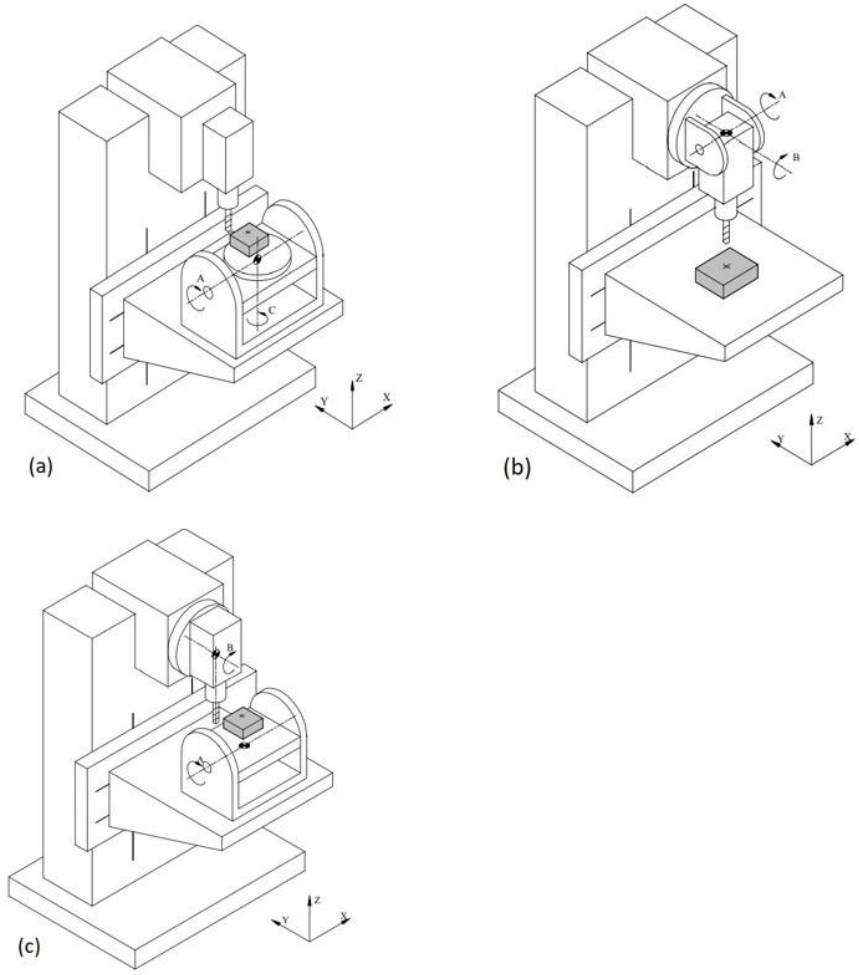

Figure 2.2 Three structural types of five-axis machine tools (a) Rotary table(b) Spindle rotating (c) Hybrid [16]

2.2 Kinematic Chains for Motion Simulation

A robot usually consists of a series of links. These links are connected by joints with one degree of freedom. The pose of the end link can be calculated if all joint movements which include linear translation movements and rotational movements are known. Pose of a link indicates the location and orientation of the link. A kinematic chain contains the configuration and the relationship between joints and links of the given robot, so it is core information of motion simulation for robots. Denavit-Hartenberg (D-H) convention is widely used by roboticists to describe kinematic chains and calculate the pose of each link of robots [19].

URDF is a file in XML format that describes a robot and it details the information of links, joints, dimensions, and so on. It can be used to model a robot with links connected by joints in a tree structure. Most industrial robots can be modeled by chains of joints with links. URDF is a fundamental robot model file in ROS, which is a flexible framework for developing robot software. It is similar to D-H convention in many respects, but with significant additional enhancements. For instance, URDF does not require coordinate systems of links that can only rotate about the Z axis. It uses an arbitrary axis for revolute (rotary) or prismatic (translational) joints [20]. Also, D-H convention does not specify the geometric models of robots. URDF specifies robot models using primarily two language elements, namely links and joints which are shown in Figure 2.3. A link describes a rigid body part of a robot by specifying its origin, mass, inertia, geometry, and texture, whereas a joint describes the connection of two links. The joint specification includes how two of links connected by the joint can move relative with each other and the movement limits [21]. Figure 2.4 illustrates an example of URDF of Link 1, Link 2, and Joint 1 in Figure 2.3.

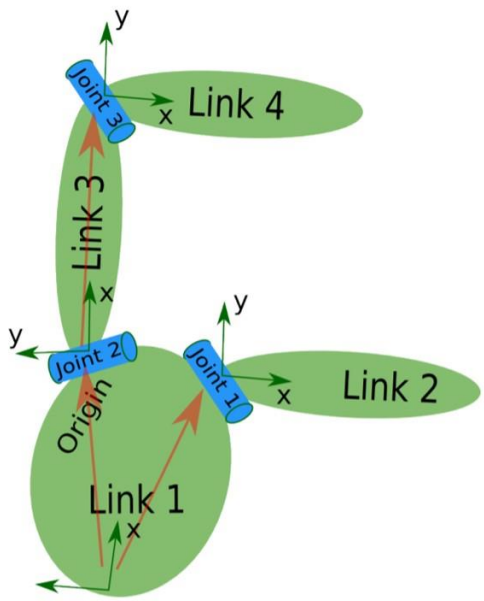

Figure 2.3 Visualization of URDF's basic language elements: links and joints [22]

```
<link name="link1">
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://example/meshes/link1.STL"/>
    </geometry>
    <material name="Cyan">
      <color rgba="0 1.0 1.0 1.0"/>
    </material>
  </visual>
</link>
<link name="link2">
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://example/meshes/link2.STL"/>
    </geometry>
    <material name="Cyan">
      <color rgba="0 1.0 1.0 1.0"/>
    </material>
  </visual>
</link>
<joint name="Joint1" type="revolute">
  <origin xyz="-1 0 1"rpy="0 0 3.1416"/>
  <parent link="link1"/>
  <child link="link2"/>
  <axis xyz="0 1 0"/>
  <limit effort="30" velocity="1.0" lower="-1.5707" upper="1.5707"/>
  </origin>
</joint>
```

Figure 2.4 URDF example

URDF of a simple robot model can be manually created using an XML editor. However, it is not practical to work with complex models using the editor. In a literature proposed by Kang et al. [23], they presented a system to automate the generation of URDF files that only requires the parameters of D-H convention. GUI was created to help users to insert the necessary inputs for the system. Nonetheless, obtaining parameters of D-H convention requires a lot of work and knowledge such as getting offset distance and angle difference between coordinate frames of robots following the convention. In addition, the presented method doesn't use the real geometric model of each link. Instead, the geometry of each link is arbitrarily set to a box shape.

The Solidworks [24] plugin called URDF Exporter has been developed by the ROS community. Solidworks which is a computer-aided design (CAD) and computer-aided engineering (CAE) computer program. The URDF Exporter can convert Solidworks models of robots into URDF with its interactive CAD environment. In [25-27], exploiting URDF Exporter to generate URDF and the geometric model for each link of robots for motion simulation was demonstrated. URDF Exporter exploits the interactive features provided by Solidworks with CAD models to help users export URDF. In order to generate URDF, the configuration tree of a robot, the coordinate system of each link, joints type, joint transforms and axes, and so on are necessary. Also, the geometric model of each link that might consist of multiple link components is also required. The link components that constitute a link and locations of rotary axes of the rotary joints can be specified manually in the interactive CAD environment of Solidworks. Although the interactive CAD environment facilitates the generation of URDF significantly, it is still a lengthy process and requires many inputs that need knowledge in robotics.

2.3 Interference Detection and Voxel Modelling

Interference has two distinct types that the first type is coincident interference and the second one is collision interference. Coincident interference indicates that two objects contact each other with surfaces. Collision interference, on the other hand, occurs when an overlapping volume exists between components. Two physical objects cannot occupy the same space at the same time but can share the same surface and it is the same in the virtual environment for simulating physical objects. Therefore, interference detection is necessary during simulation to ensure that interference between simulated objects is determined. Interference detection can be implemented for objects represented by polygons such as triangular mesh surface representation [28]. It can also be done for objects represented by voxels (volume pixel or the 3D analog of the 2D pixel).

A voxel is the cubic unit of volume which is centered at the grid point in a given space. Volumetric data (x, y, z, v) represents the value v of certain property that is stored in the voxel centered at (x, y, z). Voxels take at regularly spaced intervals along three orthogonal axes in the space which means each voxel has the same height, width, and length. Since voxel is defined on a regular grid, a 3D array (also called voxel space) is typically used to store the values with locations of voxels on the grid [29, 30]. Voxelization is a process to convert geometric objects into a set of voxels in the voxel space. The polygonal model of the object (that use polygons to represent the surface of the object) is placed in the voxel space, where at the beginning the value of each voxel is set as empty (v = 0). In the case when polygons of the object intersect with voxels, these voxels are labeled as surface voxel or full (v = 1). The surface voxels form the voxel model of the object (Figure 2.5).

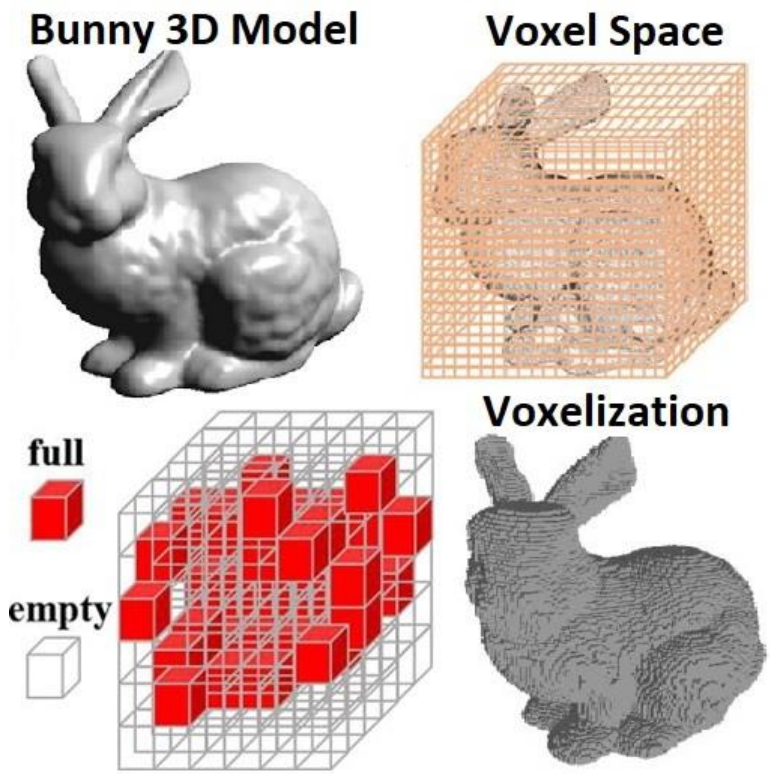

Figure 2.5 Voxelization Steps [31]

Interference between objects occurs when there is a voxel in the voxel space is labeled as a surface voxel by more than one object. In order to increase the efficiency of voxelization, Sagardia et al. [32] used two phases method to get surface voxels of objects. The first phase is to find the potential surface voxel and the second phase uses the Separating Axis Theorem, in a similar way as explained in [33], to detect interference between triangle and potential surface voxels. Lock et al. [34] mentioned that interference detection using polygon representation is computationally expensive and therefore slow. The voxel-based approach for interference detection exploits computer memory to increase efficiency. It allows faster interference detection but requiring additional memory to store the occupancy information.

2.4 Generation of Kinematic Chains of Multi-Axis Machine tools

Multi-axis machine tools can be considered to be a special type of robot with configuration constraints. It requires fewer inputs to generate kinematic chains compared with URDF Exporter designed for general robots with infinite configurations. In the articles of Lin et al. [35] and Suh et al. [36], they worked on constructing environments to simulate three-axis machine tools with kinematic chains. In order to generate kinematic chains, the 4×4 homogeneous transformation matrices between each pair of links are input manually. Mei et al. [13, 37] constructed an environment that is able to simulate five-axis machine tools with the kinematic chains for machining simulation and detection of collision between links. Kinematic chains used in this work are represented in tree-shaped charts that are generated by transformational matrices input by users. Although these works presented methods to create kinematic chains for motion simulation, they didn't provide ways to help users to avoid breaking configuration constraints of multi-axis machine tools and to select the geometric models for each link.

Chang et al. [38] and Lin et al. [39] developed universal environments for 5-axis virtual machine tools that come with user interfaces to assist users in specifying configurations to avoid breaking configuration constraints and selecting the geometry of each link of machine tools to generate kinematic chains. Two user interface dialogs are shown in Figure 2.6 and Figure 2.7. The environments require users to input parameters of D-H conventions through the user interface for motion simulation. Although their works exploit the configuration constraints of 5-axis machine tools to shorten the process to generate kinematic chains, obtaining parameters of D-H conventions requires lots of works as mentioned. Yang et al. [40] presented a generalized kinematics model using screw theory for configurations of all five-axis machine tools. Screw theory allows a global

description of rigid body motion, so that there is no need to build local coordinate frames on each drive module which is required by D-H conventions. Also, the built kinematic chain is easier adapted to other configurations. However, the geometric model of each link in these works doesn't consider to be formed by multiple link components. If each link consists of multiple link components, it has to be grouped manually by users which is a lengthy process and requires users to have knowledge in machine tool design. Unfortunately, there is a lack of work that considers automatic grouping link components and generating kinematic chains for multi-axis machine tools.

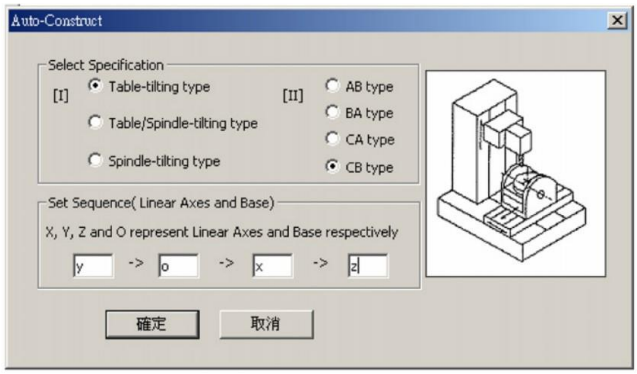

Figure 2.6 Configuration selection dialog [39]

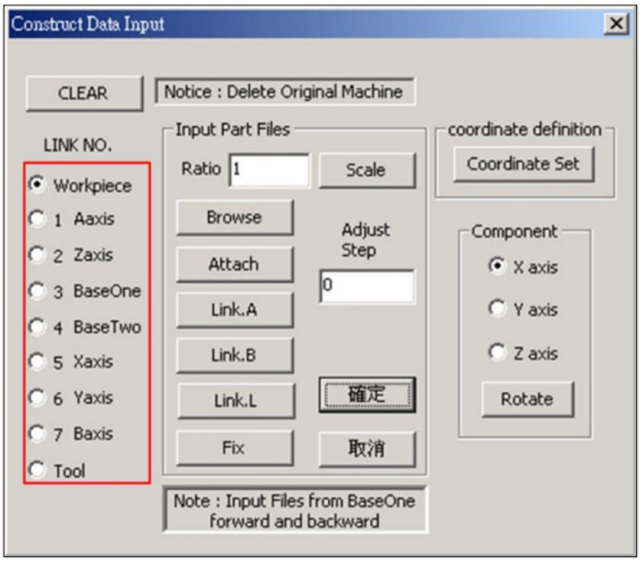

Figure 2.7 Assembly dialog [39]

Chapter 3: Methodology

In this chapter, the purpose and required inputs of each tool used in this work are discussed. In addition, the workflow of the automatic grouping of link components to generate kinematic chains of multi-axis machine tools is presented.

3.1 Required Inputs

The flowchart shown in Figure 3.1 illustrates the required inputs and outputs of each tool used for link-component grouping and generating kinematic chains in this work.

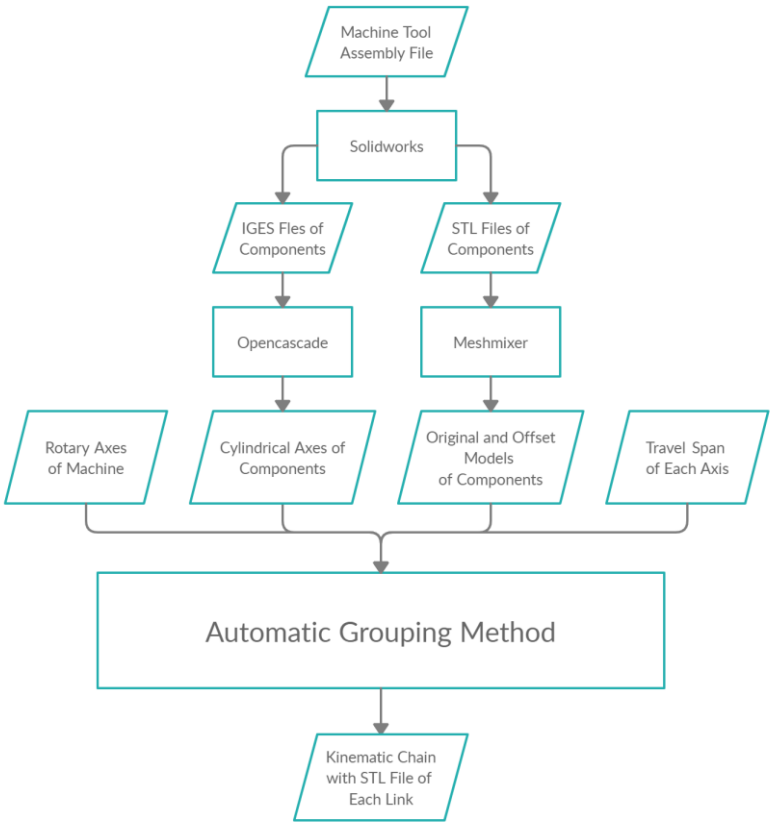

Figure 3.1 Flowchart of inputs and outputs

Three basic inputs for automatic grouping method are required from users which are the assembly file, rotary axes, and travel span of each axis of a given machine tool. These basic inputs can be obtained from the machine tool builder of the machine. Rotary axes and the travel span of each axis of a machine tool are input directly into the automatic groping method. The assembly file of a machine tool, on the other hand, is processed by three tools first. It is input into Solidworks which can read assembly files and attain geometric models and poses of components. After reading the assembly file, Solidworks exports two different formats to represent geometric models of the machine tool. These two formats are then input into two tools respectively.

The first format export from Solidworks is STL format which is a triangular mesh representation of 3D models. STL files are then inputted in Meshmixer [41] to be offset inwards. Meshmixer is chosen to offset 3D models in this work because it is developed by Autodesk which is a prestigious company developing CAD software. It is user-friendly and provides a robust function to offset mesh models by a given distance. The offset models are used to distinguish coincident and collision interference and it is discussed in chapter 4. When the distance of offsetting inwards relatively too large for a triangular mesh model, it returns an empty model without any mesh after offsetting. In this work, if a triangular mesh exists after offsetting by Meshmixer, output the offset triangular mesh model into the automatic grouping method. If it is an empty model, then output the original triangular mesh from Solidworks alternatively.

The second format is Institute for Global Environmental Strategies (abbreviated as IGES) which is one of boundary representation (abbreviated as B-rep) formats. Because triangular mesh models such as STL format use triangle mesh to represent 3D models, geometric features such as arcs and

lines in the models are not accessible. B-rep preserves the underlying geometry such as surfaces, curves, points, cylinders, and so on. Thus, it allows richer geometric operations such as finding cylindrical axes of cylinders in a 3D model. IGES file of each component are then input into Open Cascade [42]. Open Cascade is an open-source B-rep modeling toolkit and is written in C++. It can be easily integrated into the environment of this work. With its powerful functions of B-rep operation, it is one of the most popular toolkits in CAD. Open Cascade is integrated into this work so that it can read B-rep models and attain all cylindrical features of components. If the most common cylindrical axis of the cylindrical features of each component along a given direction exists, store them with the corresponding component. The cylindrical axis direction depends on what rotary axis a given machine tool has. For instance, for a five-axis machine tool that has B and C rotary axes, the most common cylindrical axis in Y and Z direction has to be collected for each component. It is important to note that in a kinematic chain, rotary axes of rotary joints are the essential information to indicate how links rotate relatively to each other. The cylindrical axes of components obtained by Open Cascade provide the information to attain the rotary axes of joints. Moreover, these cylindrical axes are the essential information to identify LIPs from CCPs which is presented in the next chapter.

3.2 Automatic Grouping Method

The presented automatic grouping method incorporates three modules which are link-component grouping, validation of link groups, and resolution of link collision with four required inputs. The assembly file of a given machine tool is processed to get the information of cylindrical axes, original models, and offset models of components first before inputting into automatic grouping method. A flowchart that includes the workflow of these three modules is depicted in Figure 3.2. The link-component grouping module uses the contact relationship between components to get link groups. The contact relationship between components in this work are generated by interference detection using voxel modeling. Voxel modeling is chosen for interference detection in this work. It is because interference detection with multiple objects using triangular mesh modeling requires checking all triangle-triangle intersection between a pair of objects and all pairs of objects have to be checked. In comparison, voxel modeling exploits computer memory to store the occupancy of a set of voxels that are labeled as filled (surface voxel) or empty in the voxel space to detect interference. Only triangle-voxel intersection checking between triangles of objects and voxels in the voxel space (voxelization for each object) is necessary. Therefore, interference detection using voxel modeling has a higher potential to be faster than using triangular mesh modeling.

After link-component grouping, it can be recognized that whether an uncertainty exists in link groups. Certain components can be grouped into multiple links and decisions are made to assign these components to one of the links. This is where the uncertainty of link-component grouping comes from. All the potential links that these components can be assigned to are stored and are used for regrouping when the link groups are incorrect.

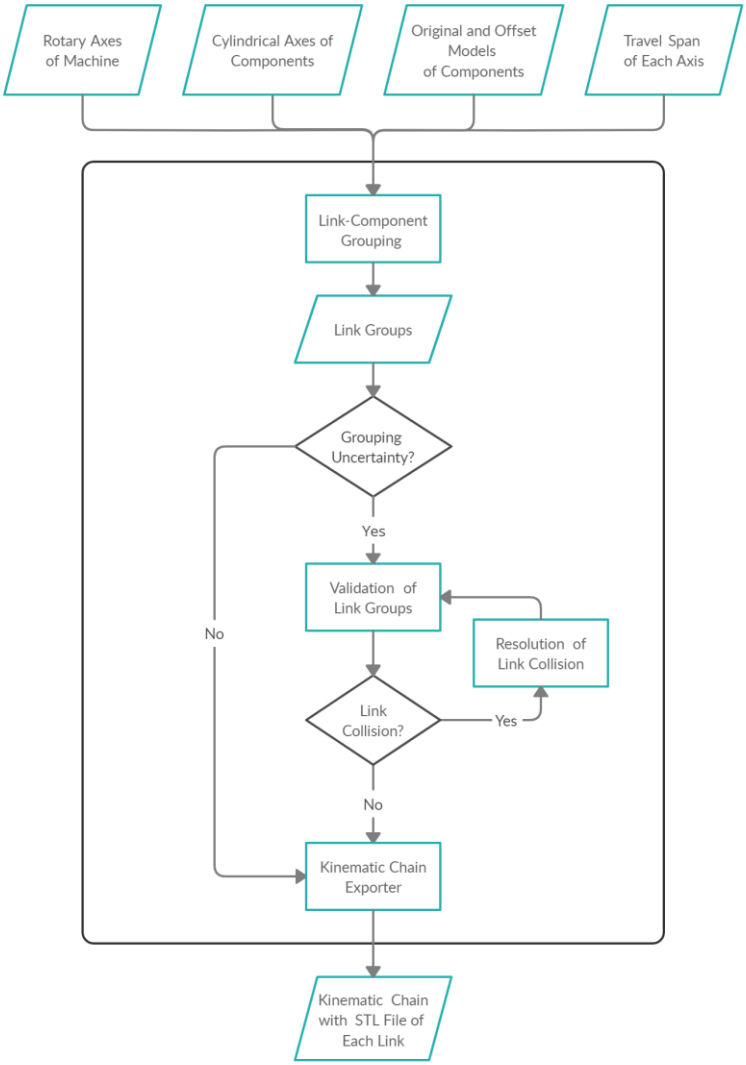

Figure 3.2 Automatic grouping method flowchart

If there is no uncertainty from link-component grouping, the exported link groups can be output directly to the kinematic chain exporter with other necessary inputs to generate the kinematic chain with STL file of each link of a given machine tool. If an uncertainty exists, on the other hand, the validation of link groups is mandatory to ensure the link groups are correct. As incorrect link groups result in collision between links, which is also called link collision, when each link of the machine tool translates or rotates within its travel span, collision information between links can be

used for the validation of link groups. On the condition of no link collision, the link groups can be exported to the kinematic chain exporter for outputting the kinematic chain with the STL file of each link of the machine directly. However, in the case when link collision exists, it indicates link groups are incorrect. Resolution of link collision is able to reassign the components, that contain uncertainties during link-component grouping and cause link collision, into their potential links. The potential links of these components are stored during link-component grouping. Reassign these components with different combinations to generate new link groups and go through the validation of link groups again. The iterative step of validation and resolution continues until no link collision exists.

In order to generate the link groups for a given machine tool assembly and have the validation of link groups, it is essential to distinguish between two distinct types of interference which are coincident interference and collision interference. Machine Tools consist of components and these components must have interference with the neighbor components. The interference between components can only be coincident interference. Coincident interference occurs while only interference surfaces between components exist. It creates no interference volume between components when components are placed at the correct location and when two components have relative movements in the correct direction. For example, coincident interference always occurs between the bottom surface of the workpiece attached to a machine table and the top surface of the machine table when they are both static. If the workpiece is translated along the machine table surface, there is still coincident interference between them without creating interference volume. In the case when a component in the simulation environment has no interference with other

components or the virtual ground, it indicates that the machine tool assembly file exists missing components or the location of that component is incorrect.

Collision interference, on the other hand, indicates that an overlapping volume exists between components. It occurs when components are at the incorrect location or components have relative movements in the wrong direction. For instance, if the workpiece is placed at the wrong location or when the workpiece is translated downwards which is a wrong relative movement direction, it causes collision interference with the appearance of interference volume.

Chapter 4: Link-Component Grouping

In this chapter, contact-component pairs (CCPs) and link-interface pairs (LIPs) are introduced. They are defined by the contact relationships between components of a given machine tool and are the core information of link-component grouping. The algorithm of link-component grouping is also presented in this chapter.

4.1 Contact-Components Pairs

In order to generate link-component groups, contact relationships between components are essential. The assembly file of a given machine tool contains the poses of each component including the orientations and locations relative to the origin coordinate system. The geometric models of components can also be obtained from the assembly file. These can be used to obtain contact relationships between components with interference detection. All pairs of components that contact each other are collected and these pairs are called contact-component pairs (CCPs) of a machine tool. Figure 4.1 illustrates a CCP map formed by the CCPs of UMC-750 which is a five-axis machine tool built by Haas. Nodes in the map represent components of the machine tool and edges are the CCPs between components.

CCPs between components in this work are generated by interference detection using voxel modeling. Components of machine tools are voxelized following the method in [32]. First, place triangular models in the voxel space, where at the beginning each voxel is labeled as empty. For each triangle in a model, the voxels within the bounding box of the triangle are traversed and checked for intersection with the triangle. If they intersect, these voxels are labeled as surface

voxels. Intersection checking between triangle and voxel follows the Separating Axis Theorem explained in [33]. If one of the voxels in the voxel space is labeled by more than one component as a surface voxel, interference occurs between these components and form CCPs between them. To voxelize components, triangular mesh models of components in STL format are required and these STL files can be obtained from Solidworks after reading the assembly file of the given machine tool.

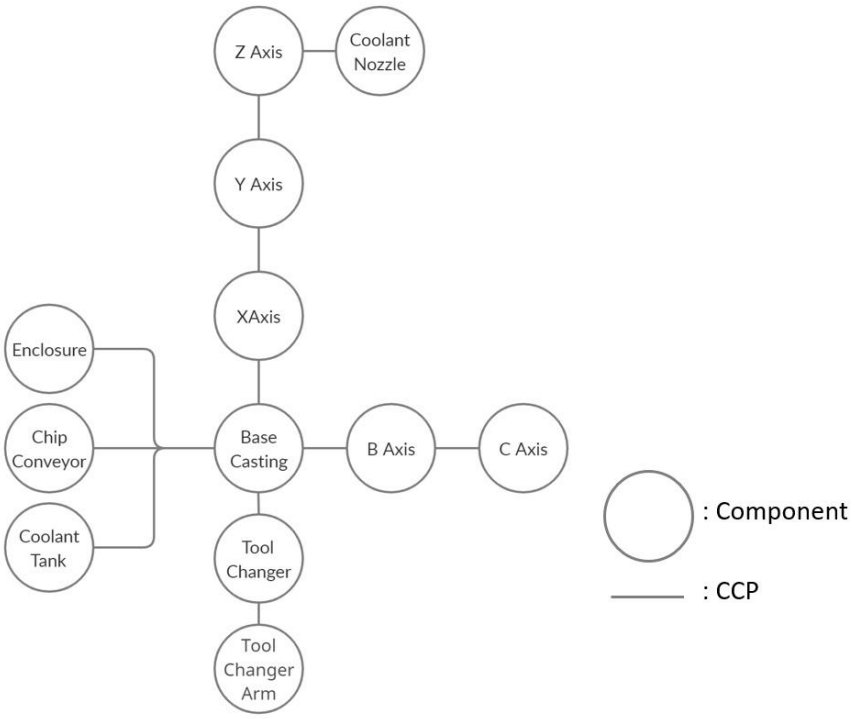

Figure 4.1 CCP map of UMC-750

It is important to note that due to the nature of voxel modeling, the accuracy of voxel modeling relies on voxel size. False interference can appear if the voxel size is larger than the distance between components that do not contact each other. An example of false interference is demonstrated in Figure 4.2. False interference results in false CCPs and false CCPs can increase

the chance that an uncertainty exists in link groups. The method to handle the uncertainty of link-component grouping is discussed in chapter 5.

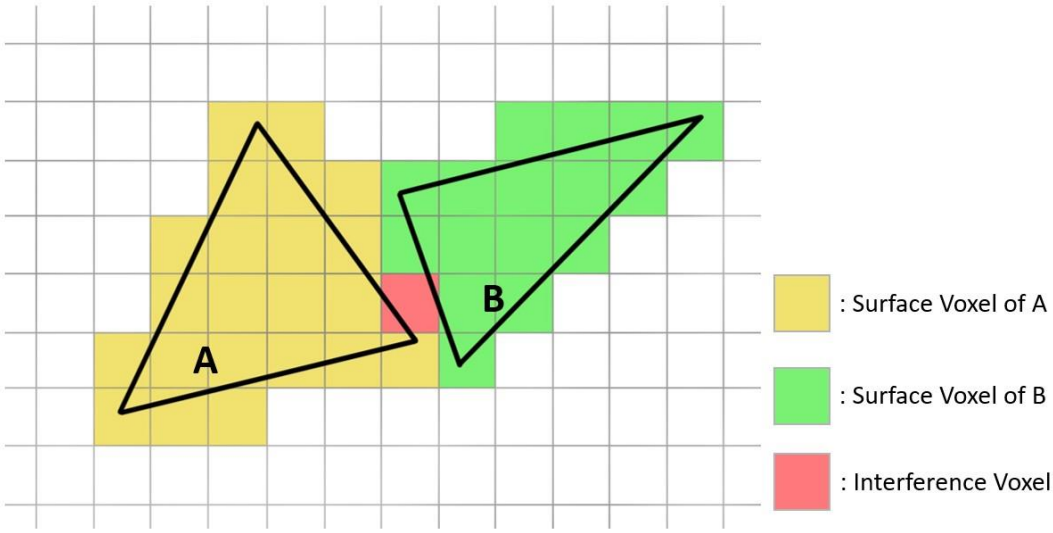

Figure 4.2 False interference in 2D

4.2 Link-Interface Pairs

With the information about how components of a machine tool contact each other, the next step is to determine which CCPs are the interfaces between link groups. These CCPs are called link-interface pairs (LIPs). LIPs are important for generating link-component groups of a machine tool because they specify where the boundaries of link groups are in the CCPs map. They are equivalent to the joints of two connected links of a machine tool. There are two types of LIPs which are: translational LIPs and rotary LIPs. Within translational LIPs, there are X-LIP, Y-LIP, and Z-LIP that represent the interfaces between link groups that can have relative translational movements in the direction of X, Y, and Z axis respectively. As for rotary LIPs, it consists of A-LIP, B-LIP, and C-LIP. They represent the interfaces between link groups that can have relative rotational

movements about a specific axis and this axis is aligned with X, Y and, Z axis respectively. In Figure 4.3, it depicts a CCP map with five LIPs of UMC-750. Red edges indicate LIPs and the alphabet of each red edge shows the type of LIP.

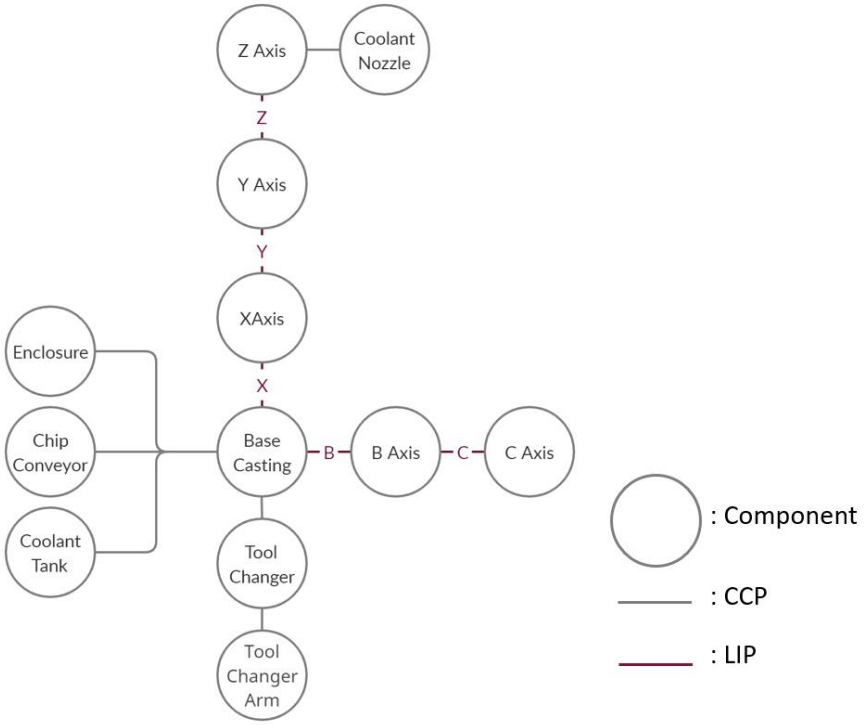

Figure 4.3 CCP map with LIPs of UMC-750

Different LIPs have distinct properties that are inherited from the corresponding joint. These properties can identify which CCPs are LIPs and the type of LIP by interference detection using voxel modeling. It is essential to distinguish coincident interference from collision interference and false interference because the collision interference information between two components in CCPs after relative movement can be used to identify LIPs from CCPs. Two components in a CCP

always have interference between each other, and it could be coincident interference, collision interference, or false interference when voxel modeling is used for interference detection.

Distinguishing collision interference from others can be done by using offset-inwards voxel models. Offsetting voxel inwards can guarantee that interference occurs between voxel models of components is collision interference. To get offset voxel models of components in order to identify LIPs, the triangular mesh models of components are offset inwards first and then are voxelized into voxel models. When the size of the component is smaller than the offset distance, the triangular mesh model of the component disappears after offsetting. It causes a problem that this component cannot be voxelized for interference detection. Thus, a hybrid method for voxelization is used to do interference detection. If a component doesn't disappear after offsetting, the offset triangular mesh model is used for voxelization, otherwise, the original triangular mesh model is used.

The triangular mesh models are offset inwards by a distance of $\sqrt{3} \times (\text{Voxel size})$. $\sqrt{3} \times (\text{Voxel size})$ is the diagonal length of a voxel and the minimum distance that guarantees the interference between voxel models is collision interference when at least one of the voxel models, that cause the interference, is offset. In order to offset triangular mesh models of components, Meshmixer is used to offset the triangular mesh models of components in this work. Meshmixer also offers the function to reduce mesh number that still well preserves shapes of geometric models, so fewer triangles in the triangular mesh models have to be voxelized and the computational time can be reduced. The triangular mesh models of components in UMC-750 before and after offset $4\sqrt{3}$ mm inwards by Meshmixer are shown in Figure 4.4 - Figure 4.6. They

demonstrate that Meshmixer is able to offset models with the given offset distance. It can be noticed that certain features that are smaller than the offset distance in the models disappear after offsetting. Therefore, the offset distance cannot be too large otherwise the collision interference between offset voxel models can fail when real collision between original models occurs. The voxel models of original and offset mesh models of components in UMC-750 following the voxelization method in [32] with 4 mm voxel size are demonstrated in Figure 4.7 - Figure 4.9.

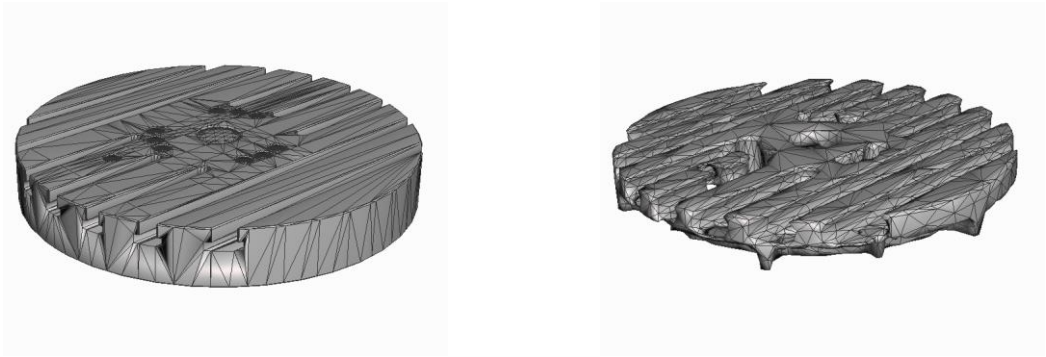

Figure 4.4 Original (left) and offset triangular mesh model of C-axis of UMC-750 (right)

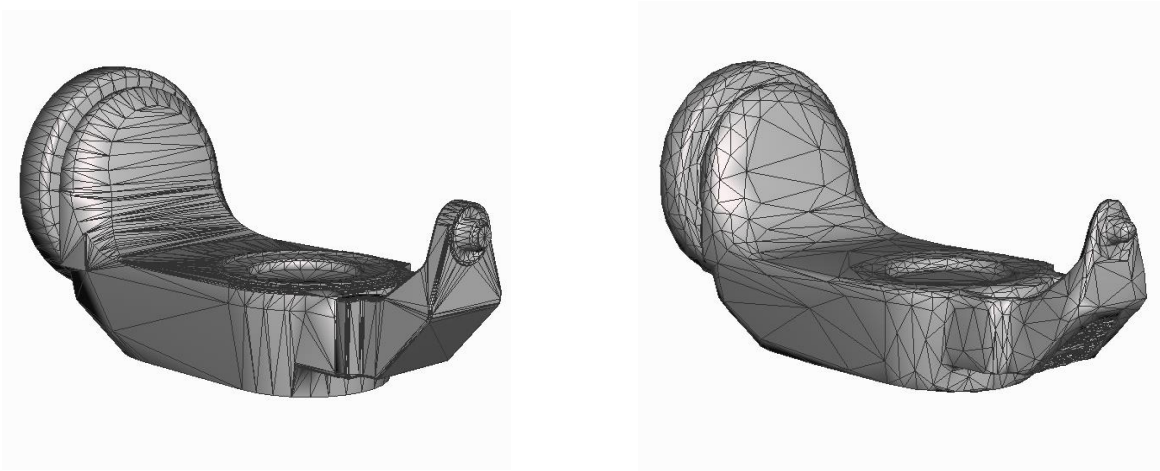

Figure 4.5 Original (left) and offset triangular mesh model of B-axis of UMC-750 (right)

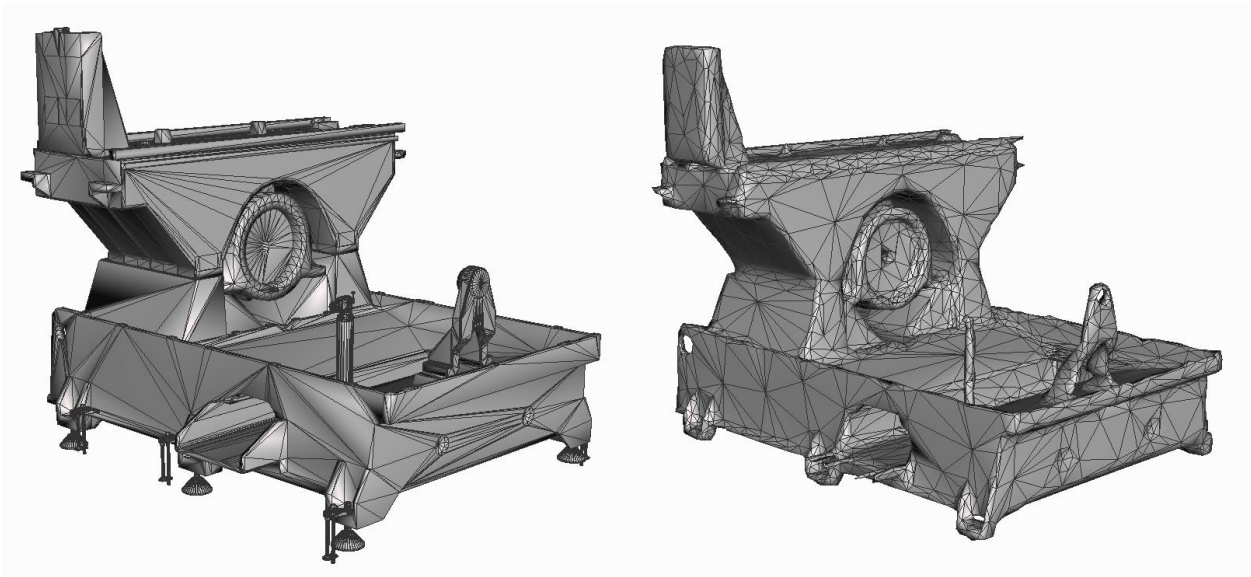

Figure 4.6 Original (left) and offset triangular mesh model of machine base of UMC-750 (right)

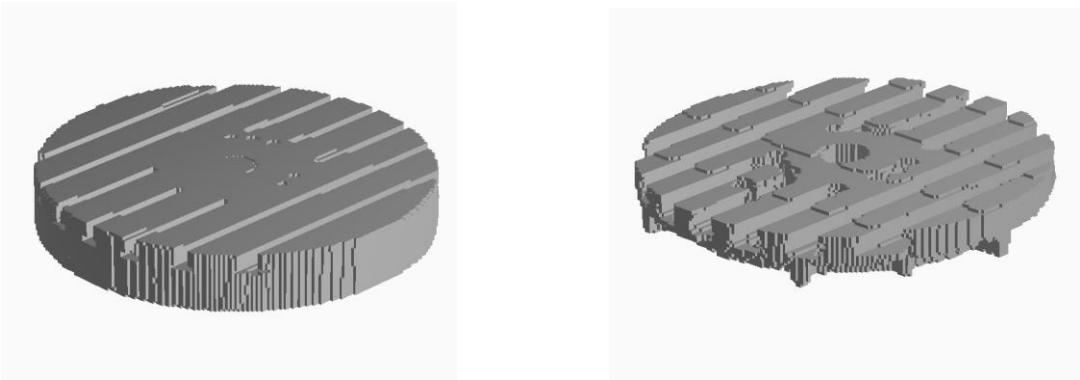

Figure 4.7 Original (Left) and offset voxel model of machine of C-axis of UMC-750 (right)

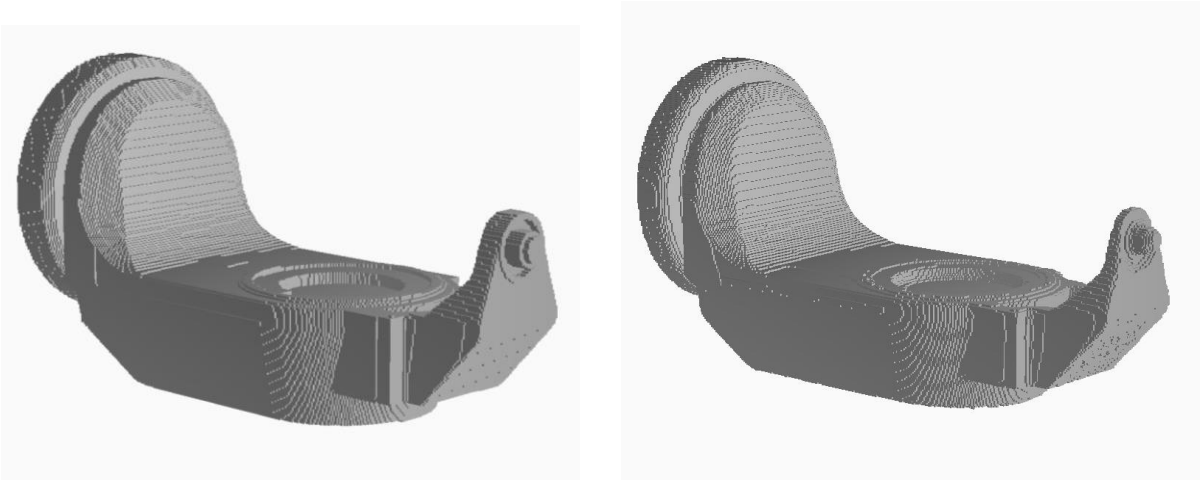

Figure 4.8 Original (Left) and offset voxel model of machine of B-axis of UMC-750 (right)

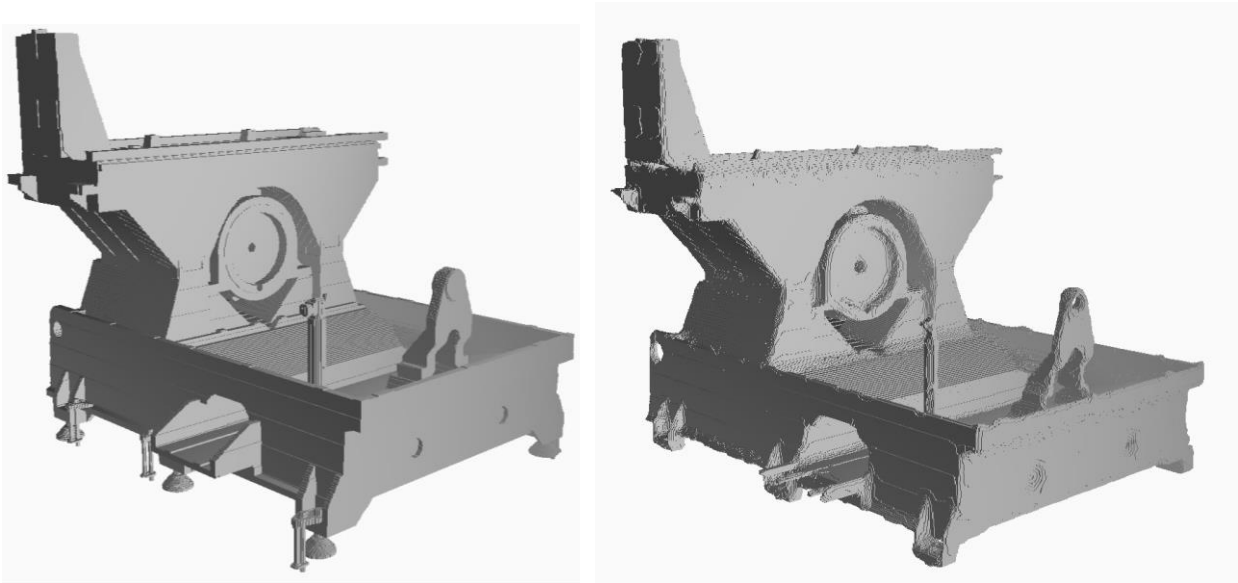

Figure 4.9 Original (Left) and offset voxel model of machine base of UMC-750 (right)

Collision interference between components cannot occur in a well-designed machine tool before anything is added to the machine. Thus, a correct link-component grouping of the machine tool assembly file cannot have collision interference while each axis moves in the correct way within the travel span. Since LIPs are equivalent to the joints of two connected links of a machine tool, the collision interference information between components of each CCP can be used to identify LIPs from CCPs with the properties of LIPs inherited from the corresponding joint. The collision interference information can be obtained by moving two components in each CCPs relatively by five out of six movements. These six movements are listed below. The rotary axes of a machine are known because they are the required inputs of the presented method from users. Therefore, the collision interference information of only two rotational movements in the A, B, or C axis is necessary depended on what rotary axes the machine tool has for five-axis machine tools. For four-axis machine tools, the Collision interference information of solely one rotational movement in the A, B, or C axis is necessary depended on what rotary axis the machine tool has. The collision interference information of a given relative movement between components is simply collision occurs or not after the movement.

1. X axis linear translational movement
2. Y axis linear translational movement
3. Z axis linear translational movement
4. A axis rotational movement
5. B axis rotational movement
6. C axis rotational movement

Collision interference detection between two components of a CCP for linear translational movements is done by fixing one of the components and translating the other one in both positive and negative direction of the given axis by the distance of $2 \times (\text{voxel size}) \times \sqrt{3}$. Therefore, there is collision interference information in the positive and negative direction of X, Y, and Z axis. Since LIPs represent interfaces between link groups, both components in each LIP has coincident interference and $2 \times (\text{voxel size}) \times \sqrt{3}$ is the distance of two offset distance. In this work, the components that have no offset voxel models after offsetting are assumed that they are not the potential components of LIPs. Also, the components that have offset voxel models after offsetting are assumed that the offset voxel models preserve the contact features for identifying LIPs.

A rotary axis is necessary for collision interference detection with rotational movements. This work assumes that components of rotary LIPs must contain cylindrical features, and the most common cylindrical axis of these cylindrical features in the given direction is the rotary axis of the component. In order to get rotary axes, models of components in IGES format are used because IGES format preserves the underlying geometry. Cylindrical axes of cylindrical features of models in the given direction can be identified from IGES files by Open Cascade.

Two components that are designed to have a relative rotational movement are assumed to have a common cylindrical axis along the axis of rotational movement. If they do not have a common cylindrical axis, they cannot be a rotary LIP. Therefore, it is only necessary to get collision interference information of rotational movements for the CCPs that both of their components contain a common cylindrical axis. For instance, if a five-axis machine tool that has A and C rotary axes, collision interference detection for rotational movement is necessary only if two components

in a CCP contain a common cylindrical axis in the X axis or Z axis. The angle of rotational movement for collision interference detection is set to an angle within both travel spans of the rotary axes of the given machine tool. Furthermore, the angle cannot be smaller than 10° since a small angle can miss the detection of real collision between components. For example, the travel span of B axis of UMC-500 is 120° to -35° and the travel span of C axis is 360°. The angle of rotational movement for collision interference detection can be set to -35°. After this step, all CCPs have 6 collision interference information of the X, Y, and Z axes in both positive and negative directions. For the CCPs whose two components have a common cylindrical axis, they have one more extra collision interference information.

With the collision interference information between components in each CCP, different LIPs can be identified from CCPs according to the properties of the corresponding joint of a machine tool. The criteria of different types of LIPs are listed below:

Table 4-1 Criteria of translational LIPs

X-LIP:	Y-LIP:	Z-LIP:
1. No collision occurs when one of the components in the CCP is translated in the positive and negative direction of X axis by a distance of $2 \times (\text{voxel size}) \times \sqrt{3}$. 2. Collisions occur when one of the components in the CCP is translated in the positive or negative direction of Y and Z axis by a distance of $2 \times (\text{voxel size}) \times \sqrt{3}$.	1. No collision occurs when one of the components in the CCP is translated in the positive and negative direction of Y axis by a distance of $2 \times (\text{voxel size}) \times \sqrt{3}$. 2. Collisions occur when one of the components in the CCP is translated in the positive or negative direction of X and Z axis by a distance of $2 \times (\text{voxel size}) \times \sqrt{3}$.	1. No collision occurs when one of the components in the CCP is translated in the positive and negative direction of Z axis by a distance of $2 \times (\text{voxel size}) \times \sqrt{3}$. 3. Collisions occur when one of the components in the CCP is translated in the positive or negative direction of X and Y axis by a distance of $2 \times (\text{voxel size}) \times \sqrt{3}$.

Table 4-2 Criteria of rotary LIPs

A-LIP:	B-LIP:	C-LIP:
1. Both components contain a common cylindrical axis aligned with X axis 2. No collision occurs when one of the components in the CCP is rotated about the common cylindrical axis by a given angle 3. Collisions occur when one of the components in the CCP is translated in the positive or negative direction of X, Y, and Z axis by the distance of $2 \times (\text{voxel size}) \times \sqrt{3}$	1. Both components contain a common cylindrical axis aligned with Y direction 2. No collision occurs when one of the components in the CCP is rotated about the common cylindrical axis by a given angle 3. Collisions occur when one of the components in the CCP is translated in the positive or negative direction of X, Y, and Z axis by the distance of $2 \times (\text{voxel size}) \times \sqrt{3}$	1. Both components contain a common cylindrical axis aligned with Z direction 2. No collision occurs when one of the components in the CCP is rotated about the common cylindrical axis by a given angle 3. Collisions occur when one of the components in the CCP is translated in the positive or negative direction of X, Y, and Z axis by the distance of $2 \times (\text{voxel size}) \times \sqrt{3}$

Two mesh models of components of a CCP are shown in Figure 4.10. These two components of the CCP form a guideway feature in the X axis of the machine tool so this CCP should be an X-LIP. The mesh models of components are offset first and voxelized using a 4 mm voxel size into offset voxel models in Figure 4.11. Two components before translating and collision occurs between two components after translating in +Y direction by the distance of $2 \times (\text{voxel size}) \times \sqrt{3}$ are demonstrated in Figure 4.11 (a) and Figure 4.11 (b) respectively. Collision exists when component 2 is translated relative to component 1 in +Y, -Y, and +Z direction by the distance of $2 \times (\text{voxel size}) \times \sqrt{3}$. Also, there is no common axis for this CCP. Therefore, the CCP is identified as an X-LIP according to the criterion presented above.

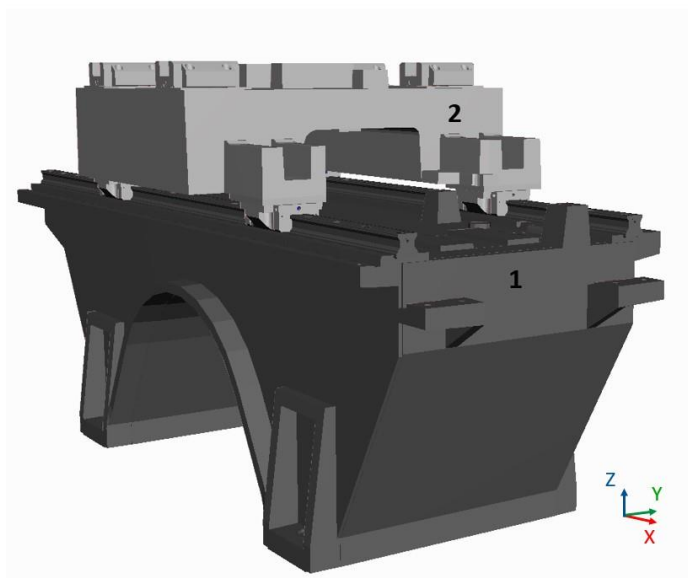

Figure 4.10 Two mesh models of components of a X-LIP

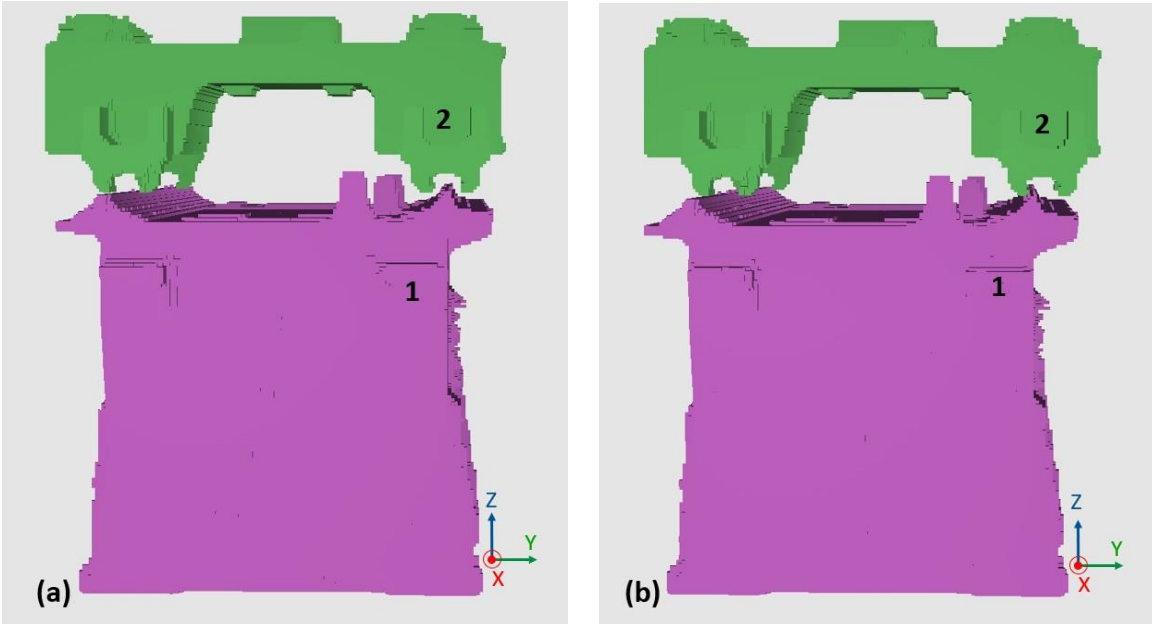

Figure 4.11 Two offset voxel models of components of a X-LIP (a) before (b) after translating

4.3 Generation of Link Groups

After having CCPs and LIPs of a given machine tool, an algorithm, which is called link-component grouping, is presented in this work to CCPs and LIPs for grouping link components of each link of a machine. It includes two steps which are growing groups from the components of LIPs and merging groups when an overlapping component between groups exists. The link groups are formed after repeating these two steps until the total group number is equal to the number of axes of the given machine tool plus one and no component, which has a connection to any other group, is left without being grouped. The total group number needs to be equal to the number of axes of the given machine tool plus one is because a machine consists of not only moveable axes but also a machine base. For each step of the presented algorithm, two rules must be followed. First, components in the same group are all connected which means each component should contact at least one component in the same group. It also indicates that each component has at least one CCP

connection with any component in the same group. Second, no LIP exists within any group. Because LIPs represent interfaces between link groups, it can only exist between link groups.

The detailed steps of the presented algorithm to generate link groups of a machine tool are listed below. The CCPs and LIPs of UMC-500 which is a five-axis machine tool built by Hass are identified using 2mm voxel size and are used to create a CCP map. The CCP map of UMC-500 in Figure 4.12 excludes some components connected with the machine base in the assembly file and is used to illustrate each step of the algorithm.

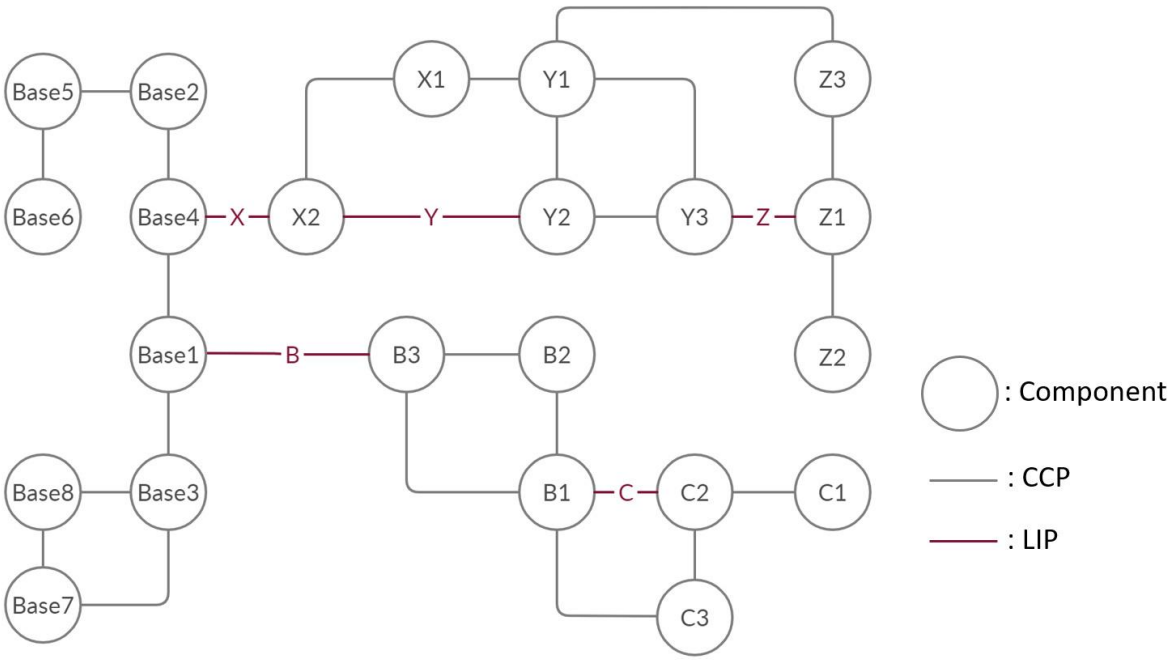

Figure 4.12 CCP map of UMC-500 example

Step 1:

Create groups starting from the components of LIPs. Assign all immediate components that are connected to the components of LIPs into a group if and only if the connection between them is not a LIP. In Figure 4.13, there are 9 groups in total after step 1.

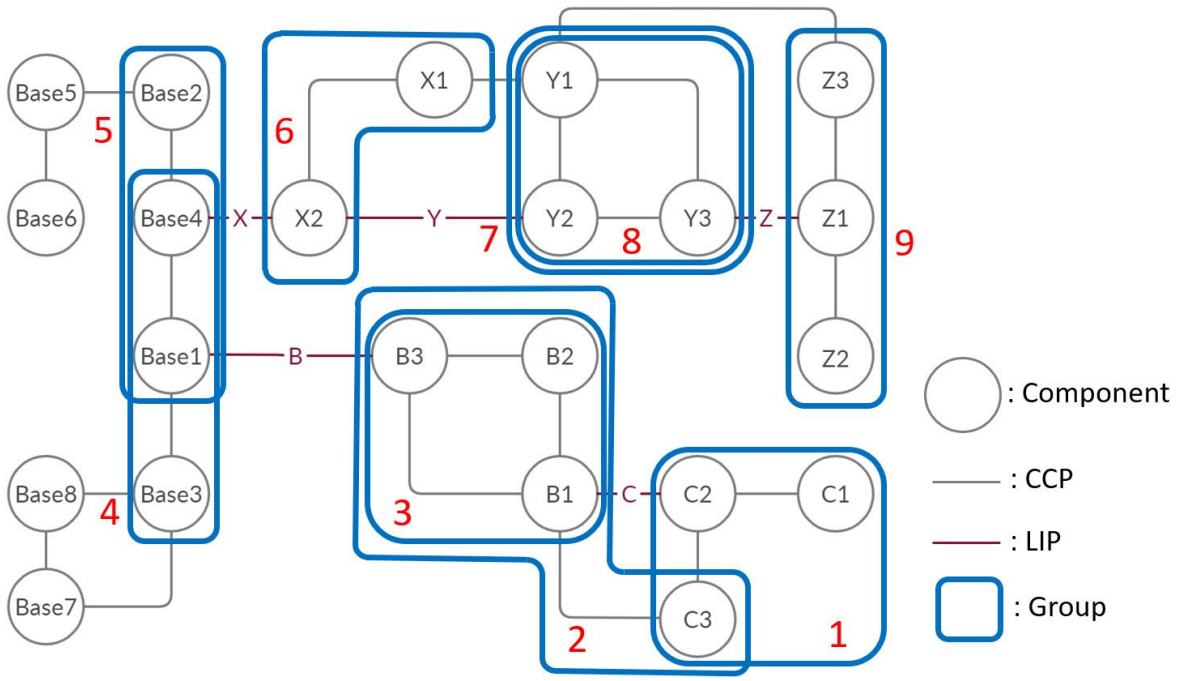

Figure 4.13 Link-component grouping of UMC-500 example after step 1

Step 2:

Merge groups that are generated from step 1 if an overlapping component between groups exists. For instance, component Base 1 and Base 4 are the overlapping components between group 4 and group 5 in Figure 4.13. Because there is no LIP after merging these two groups, group 4 and group 5 can be merged and form a new group. The same procedure can be used to merge group 2 and group 3, and group 7 and group 8 since there is no LIP in the merged group. The new group after

merging can contain a LIP. If a LIP exists within a new group after merging, skip the merging procedure for the two groups. Two strategies are presented to handle the overlapping components that cause the merged group contains a LIP. The CCP map of UMC-500 after step 2 using strategy A is shown in Figure 4.14.

Strategy A:

Assign the overlapping component, that causes a LIP to exist in the merged group, to one of the overlapping groups that contains fewer components than others. If these groups have the same number of components, assign the overlapping components to the group which is stored in the list of groups earlier than others. In Figure 4.13, component C3 is the overlapping component of group 1 and group 2. If group 1 and group 2 are merged together, the merged group contains a LIP. Therefore, these two groups cannot be merged and component C3 is assigned to group 2 that contains fewer components than group 1.

Strategy B:

Assign the overlapping component, that causes a LIP to exist in the merged group, to one of the overlapping groups that contains more components than others. If these groups have the same number of components, assign the overlapping components to the group which is stored in the list of groups earlier than others. If strategy B is used in Figure 4.13, component C3 is assigned to group 1 that contains more components than group 2.

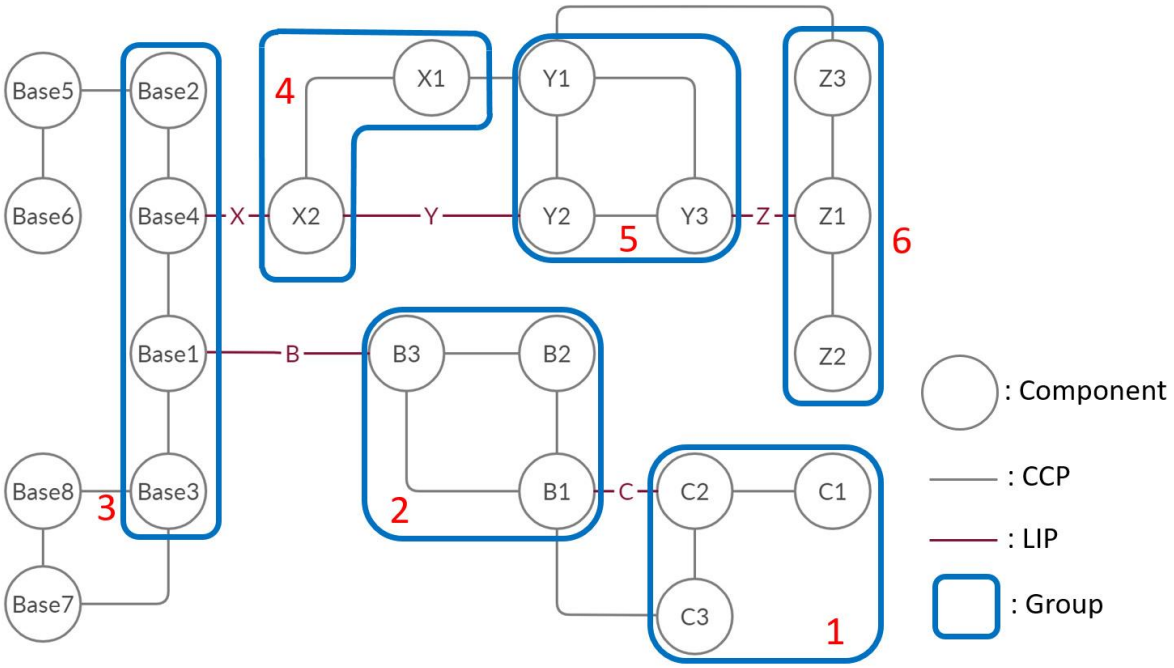

Figure 4.14 Link-component grouping of UMC-500 example after step 2

Step 3:

Repeat step 1 and 2 until the total group number is equal to the number of axes of the given machine tool plus one and no component that has a connection to any group is left without being grouped. Then, the link groups of the machine tool are generated. The link groups of UMC-500 example after link-component grouping is shown in Figure 4.15.

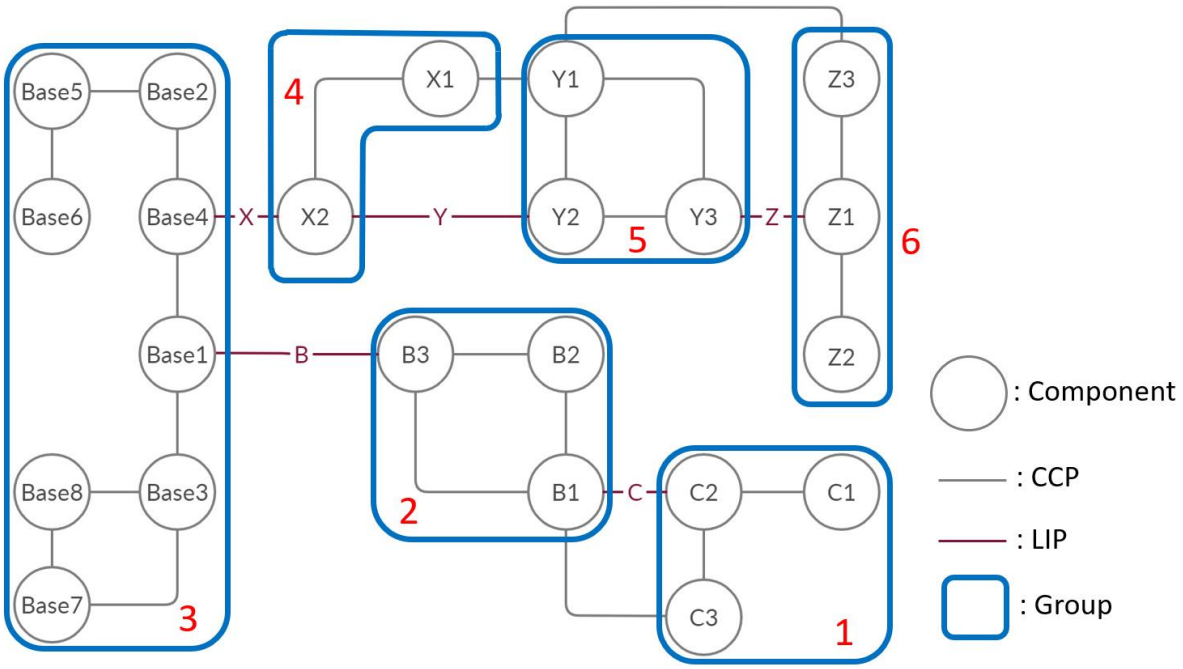

Figure 4.15 Final result of link-component grouping of UMC-500 example

The link-component grouping presented in this work assumes the number of LIPs is equal to the number of axis of the given machine tool and only one LIP exists between two link groups. When the number of LIPs is less than the number of axis of the machine tool, it implies missing components in the assembly file of the machine tool exist. It is challenging to create missing components to generate correct LIPs. Thus, the method stops proceeding with the further process if the number of LIPs is less than the number of axis of the machine tool. If the number of LIPs is more than the number of axis of the machine tool, it is required to have a more complex algorithm to generate link groups.

Chapter 5: Validation of Link Groups and Resolution of Link Collision

In chapter 4, the link-component grouping using CCPs and LIPs is discussed. The link-component grouping can create uncertainties to generate incorrect link groups. Thus, the validation of link groups is mandatory if an uncertainty exists. Link collision occurs when link groups contain incorrectness caused by the uncertainty from link-component grouping. The resolution of link collision is necessary if link collision exists to regroup the components that cause link collision.

5.1 Validation of Link Groups

There are two types of uncertainty from link-component grouping. The first type of uncertainty occurs while one of the two strategies, which are strategy A and strategy B discussed in session 4.3, is used to make decisions about assigning the overlapping components between groups. The second type of uncertainty arises when two link groups have a CCP connection after the link groups are generated. Because two link groups should be separated by only a LIP, having a CCP connection between them indicates an uncertainty exists within the link groups. Two components, which are in two different link groups and have a CCP connection between them, can be grouped into two different link group as original grouping or be grouped together into one of the link groups. For instance, Y1 and Z3 in Figure 4.15 are assigned to two different link groups by link-component grouping. However, Y1 and Z3 have a CCP connection so they both can also be grouped into group 5 or both be assigned into group 6.

These two types of uncertainty can arise from true interference and false interference between components. If a smaller voxel size is used, the chance to have the uncertainty caused by false

interference is lower. Figure 5.1 shows a CCP map of UMC-500 when 1.3 mm voxel size is used. X1Y1, Y1Y3, and Y1Z3 CCPs disappear comparing to the CCP map of UMC-500 using 2 mm voxel size in Figure 4.12. It indicates that these three CCPs are generated by false interference. It also shows that using a smaller voxel size can have fewer false interference that can cause uncertainties.

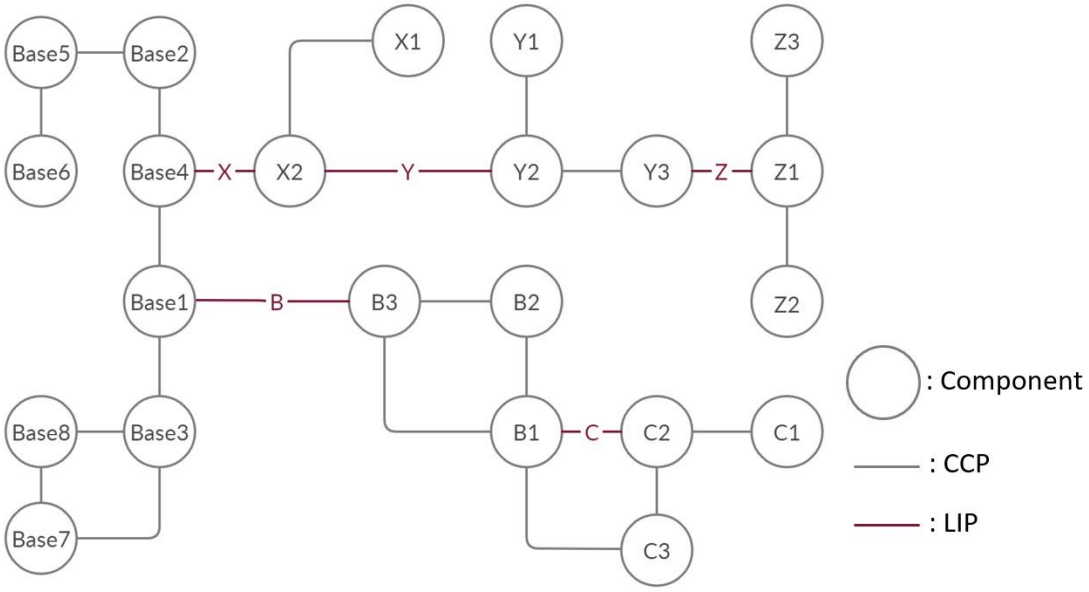

Figure 5.1 CCP map of UMC500-CCPs using 1.3mm voxel size

When an uncertainty exists, the link-component grouping can generate incorrect link groups that cause link collision when each moveable link of a machine tool moves within its travel span in the correct direction. Thus, the validation of link groups is mandatory if an uncertainty exists. It detects collision interference between links within all possible link-pose combinations of a given machine tool to identify if the link groups are correct. If no link collision arises during the validation of the link groups, link groups are export to the kinematic chain exporter to generate the kinematic chain

and STL file for each link of the machine tool. However, the resolution of link collision is necessary to regroup the component that causes an uncertainty and link collision if link collision arises during the validation of link groups. The new link groups are checked again by the validation of link groups. The iterative step of validation and resolution continues until no link collision exists and then the link groups can be export to the kinematic chain exporter.

Five-axis machine tools have six links in total which are three links of translational axes, two links of rotary axes and, and one link of the machine base. In this work, the original pose of each movable link in the assembly file of a machine tool is assumed to be zero in the corresponding travel span. Also, if the travel span of an axis is only specified by a distance, this work assumes that the axis can travel half of the travel span to the positive direction and half of the travel span to the negative direction of the corresponding axis from the zero. For instance, the travel span of the X axis of UMC-500 is 610 mm from Haas website [10]. So, the X axis can travel 305 mm to the positive direction and 305 mm to the negative direction of X axis from the origin pose defined in its assembly file.

If each movable link has N sampled poses for the validation of link groups, the travel span is divided by N evenly to get each pose of the movable link. In total, there are N^5 different link-pose combinations that need to be checked for collision interference. As a five-axis machine tool contains six links, in total $6N^5$ triangular mesh models need to be voxelized for six links at all sampled poses for the validation of link groups. However, when the link-pose combination is changed from one to the other in some cases, it is not necessary to voxelize all links again. It is because some links do not change their poses. For instance, regardless of how the link-pose

combination of a given machine tool changes, the pose of the base link never changes. It only needs to be voxelized once in order to check all link-pose combinations of the machine tool. Another example is that while changing the pose of the end link, which is the last link in the kinematic chain and only connect with one link, other links are not required to be voxelized again because their poses have no change. Therefore, it is not necessary to voxelize $6N^5$ times for validating all different poses of all links.

There are two different types of voxelization that are used in this thesis. The first type called triangle voxelization converts a triangular mesh model into a voxel model. The other one is shifting the existing voxel model in the voxel space to get a new voxel model at a different location with the same orientation and it is called shifting voxelization. Shifting voxelization only requires utilizing the existing voxel models and shift them in the voxelspace to generate new voxel models. It takes less computational time than triangle voxelization that requires looping through all triangles of triangular mesh models and checking triangle-voxel intersection to generate voxel models. However, shifting voxelization can only generate new voxel models by translating the existing voxel models without changing their orientation. If the orientation of the model change, only triangle voxelization can be applied to get the new voxel model.

The voxel models created by triangle voxelization are called parent voxel models. Parent voxel models can generate child voxel models through shifting voxelization at different sampled locations with the same orientation. According to [5], there must not be a rotary axis between any two of the three linear translational axes. Linear translational axes of a machine tool only change their location without changing their orientation. Also, voxels in the voxel space are fixed,

orthogonal, and align with the X, Y, and Z axis. For all translational axes of the machine, only one parent voxel model for each axis is needed and they can generate all their child voxel models at different sampled locations. Yet, for rotary links, triangle voxelization is necessary to generate parent voxel models for all different sampled orientations. These parent voxel models of different sampled orientations then can generate child voxel models at different sampled locations using shifting voxelization. By using shifting voxelization, the number of triangle voxelization which takes a longer computational time can be cut down dramatically in order to do collision interference detection for N^5different link-pose combinations of five-axis machine tools.

The number of necessary parent voxel models depends on the structural type and the configuration type of a machine tool. There are three structural types for five-axis machine tools which are table-rotating, spindle-rotating, and hybrid [18]. Also, five-axis machine tools can have three configuration types which are 5-0, 4-1, and 3-2. A five-axis machine tool has five axes plus one machine base in series and the machine base cuts the kinematic chain into two ends which are the spindle end and workpiece end. 5-0 configuration type indicates that five axes are in series in the same end. 4-1 configuration type has one axis in one end and four axes in series in the other end. Two axes are in one end and the other three axes are in the other end for 3-2 configuration type. Three configuration types are illustrated using tree diagrams in Figure 5.2. In Figure 5.2, T represents a translational axis (can be X, Y, or Z) and R represents a rotary axis (can be A, B or C).

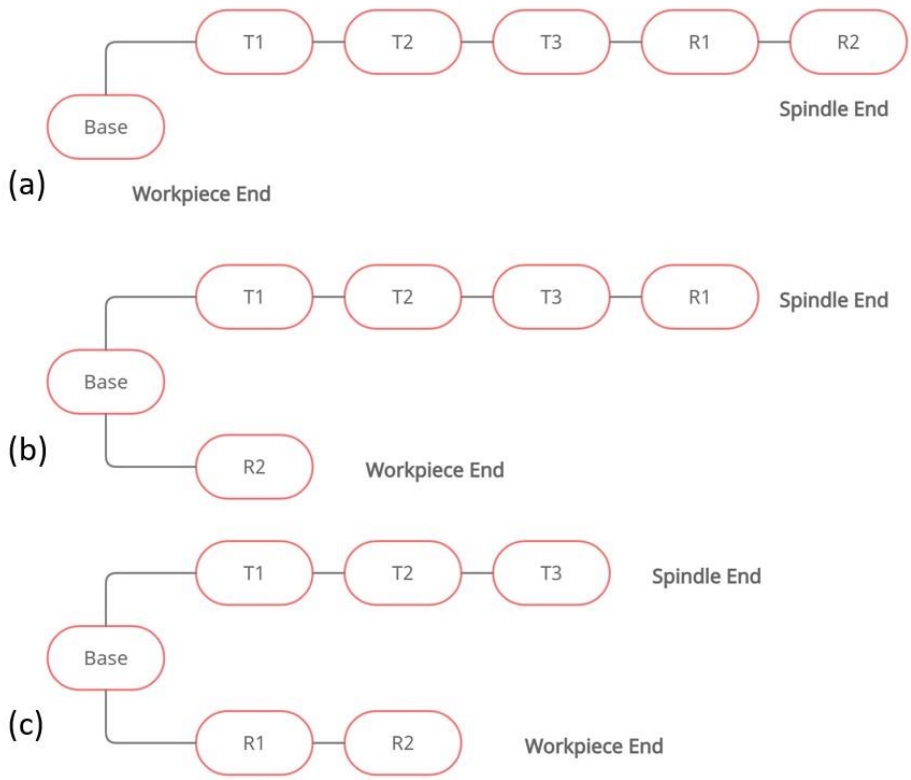

Figure 5.2 Example of configuration types of five-axis machine tool (a) 5-0 (b) 4-1 (c) 3-2

Each configuration type can have three different structural types of a five-axis machine except 5-0 configuration type that can only have table-rotating and spindle-rotating structure type. The number of necessary voxelization including triangle voxelization and shifting voxelization for checking N^5different link-pose combinations depends on different the configuration types of a given machine tool. Following the naming conventions of each link in each configuration type in Figure 5.2, the base only requires one voxelization for machine tools with all combinations of configuration types and structure type since it is always static. Starting from the base, the link next to the previous link requires N times of voxelization if each link requires N sampled poses for validation. For instance, T1 of a machine with 5-0 configuration type and table-rotating structure type requires N voxelizations. For each sampled pose of T1, the next link of T1, which is T2,

requires N voxelizations for N sampled poses of T2. Therefore, N^2 voxelizations are necessary for T2 to do the validation for all poses. The process is the same for the next link and so on. Therefore, T3, R1, and R2 requires N^3, N^4, and N^5voxelizations respectively for the validation.

The number of required parent voxel models to check N^5different link-pose combinations depends on the combination of configuration types and structure type of a given machine tool. For instance, the machine example with 3-2 configuration type and table-rotating structure type in Figure 5.2 requires only one parent voxel model for the base, T1, T2, and T3. These four parent voxel models can generate all link-pose combinations of these four links. Since shifting voxelization can only be used to generate new voxel models by translating the existing voxel models without changing their orientation, the triangle voxelization of rotary links is necessary to generate parent voxel models for all different sampled orientations. These parent voxel models of different sampled orientations then generate child voxel models at each sampled location using shifting voxelization. Therefore, R1 needs N parent voxel models for N different orientations. N parent voxel models of R2 are necessary for each orientation of R1 so there are N^2 parent voxel models required for R2.

The number of required parent voxel models for the machine tool with other combinations of configuration type and structure type can be calculated with the same concept. Four-axis machine tools can also use the same concept to calculate the number of required voxelizations and the number of required parent voxel models. The number of required voxelizations and the number of required parent voxel models for each combination of configuration types and structure type of five-axis machine tools are shown in Table 5-1.

Table 5-1 Summary of required voxelization and parent voxel model number for five-axis machine tools

	5-0 configuration type	4-1 configuration type		3-2 configuration type	
	Table-rotating or spindle-rotating structure type	Table-rotating or spindle-rotating structure type	Hybrid structure type	Table-rotating or spindle-rotating structure type	Hybrid structure type
The required voxelization number	$1\ (\mathrm{Base}) + \mathrm{N}\ (\mathrm{T1}) + \mathrm{N}^2\ (\mathrm{T2}) + \mathrm{N}^3\ (\mathrm{T3}) + \mathrm{N}^4\ (\mathrm{R1}) + \mathrm{N}^5\ (\mathrm{R2})$	$\mathrm{N}\ (\mathrm{T1}) + 1\ (\mathrm{Base}) + \mathrm{N}\ (\mathrm{T2}) + \mathrm{N}^2\ (\mathrm{T3}) + \mathrm{N}^3\ (\mathrm{R1}) + \mathrm{N}^4\ (\mathrm{R2})$	$\mathrm{N}\ (\mathrm{R2}) + 1\ (\mathrm{Base}) + \mathrm{N}\ (\mathrm{T1}) + \mathrm{N}^2\ (\mathrm{T2}) + \mathrm{N}^3\ (\mathrm{T3}) + \mathrm{N}^4\ (\mathrm{R1})$	$\mathrm{N}^2(\mathrm{R2}) + \mathrm{N}\ (\mathrm{R1}) + 1\ (\mathrm{Base}) + \mathrm{N}\ (\mathrm{T1}) + \mathrm{N}^2\ (\mathrm{T2}) + \mathrm{N}^3\ (\mathrm{T3})$	$\mathrm{N}^2(\mathrm{R1}) + \mathrm{N}\ (\mathrm{T1}) + 1\ (\mathrm{Base}) + \mathrm{N}\ (\mathrm{T1}) + \mathrm{N}^2\ (\mathrm{T2}) + \mathrm{N}^3\ (\mathrm{R2})$
The required parent voxel model number	1 (Base) + 1(T1) + 1 (T2) + 1(T3) + N (R1) + N^2 (R2)	1(T1) + 1 (Base) + 1 (T2) + 1(T3) + N (R1) + N^2 (R2)	N (R1) + 1 (Base) + 1 (T1) + 1 (T2) + 1(T3) + N (R2)	1 (T1) +1 (T2) + 1 (Base) + 1(T3) + N (R1) + N^2 (R2)	N (R1) + 1 (T1) + 1 (Base) + 1 (T2) + 1(T3) + N (R2)

5.2 Resolution of Link Collision

Incorrect link groups that result in link collision are caused by the uncertainty from the link-component grouping. The resolution of link collision is necessary only and only if link collision is detected during the validation of link groups. The first type of uncertainty occurs when one of the two strategies is used to determine how the overlapping components should be assigned. The possible groups that these overlapping components can be assigned to are stored. For the second type of uncertainty that arises when two link groups have a CCP connection, it is determined after the link groups are generated. These link groups are also collected.

A component collides with the other component in another link group can be detected during the validation of link groups. If the collided component is the component that causes the first type of uncertainty, assign the component using the other grouping strategy. If link collision due to this component still exists, assign the component to the other possible group stored during link-component grouping and do the validation of link groups until the component doesn't cause link collision. In the case when there are more than one component that cause link collision and the first type of uncertainty, assign these components with all possible combinations to the possible groups and do the validation of link groups until no link collision occurs. Using the link-component grouping example with strategy A in chapter 4, component C3 in Figure 5.4 is assigned to group 1 because group 1 contains fewer components. If component C3 causes link collision during the validation of link groups, regroup it to group 2 which is stored during link-component grouping and do the validation again.

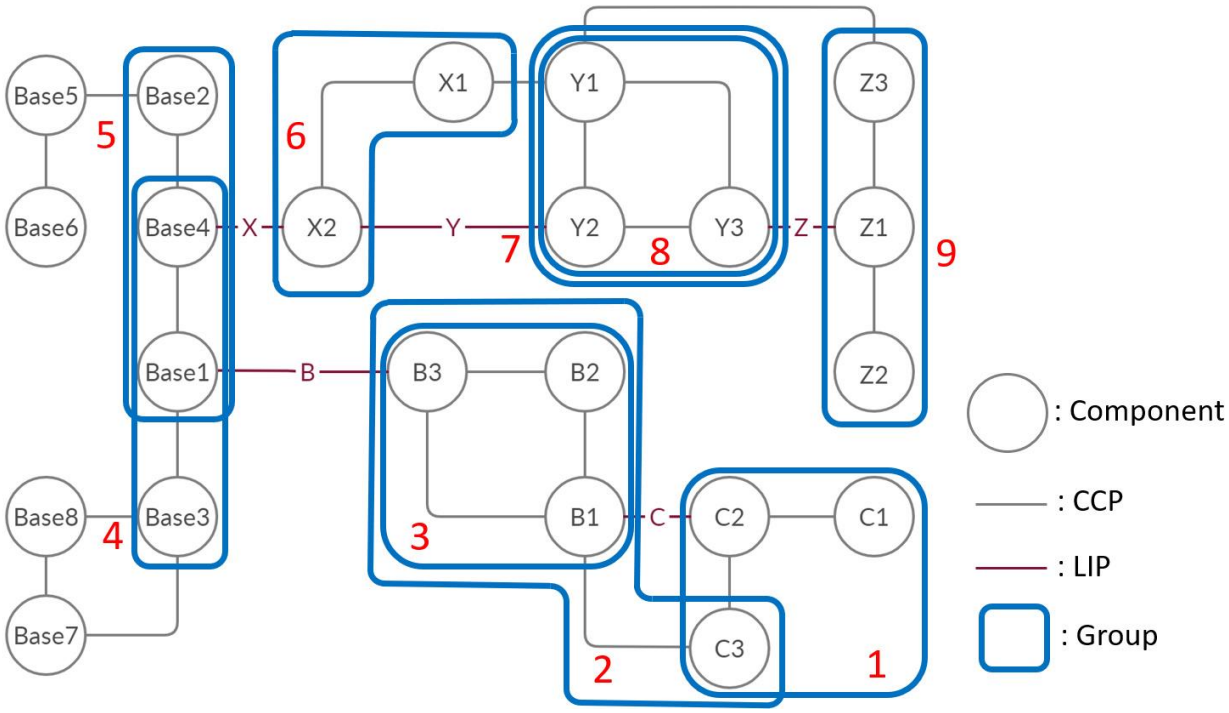

Figure 5.3 Link-component grouping of UMC-500 example after step 1

If the collided component is the component that results in the second type of uncertainty, group this component and the other component in the same CCP together into one of the groups which is connected by this CCP. In the case when link collision still exists in the new link groups, group these two components into the other groups that is connected by the CCP. Component Y1 and component Z3 in Figure 5.4 are assigned to two different link groups by link-component grouping but they are connected by a CCP. If one of these two components leads to link collision during the validation of link groups, group both of them to group 5 or group 6 and do validation again. If link collision still exists, group them into the other group.

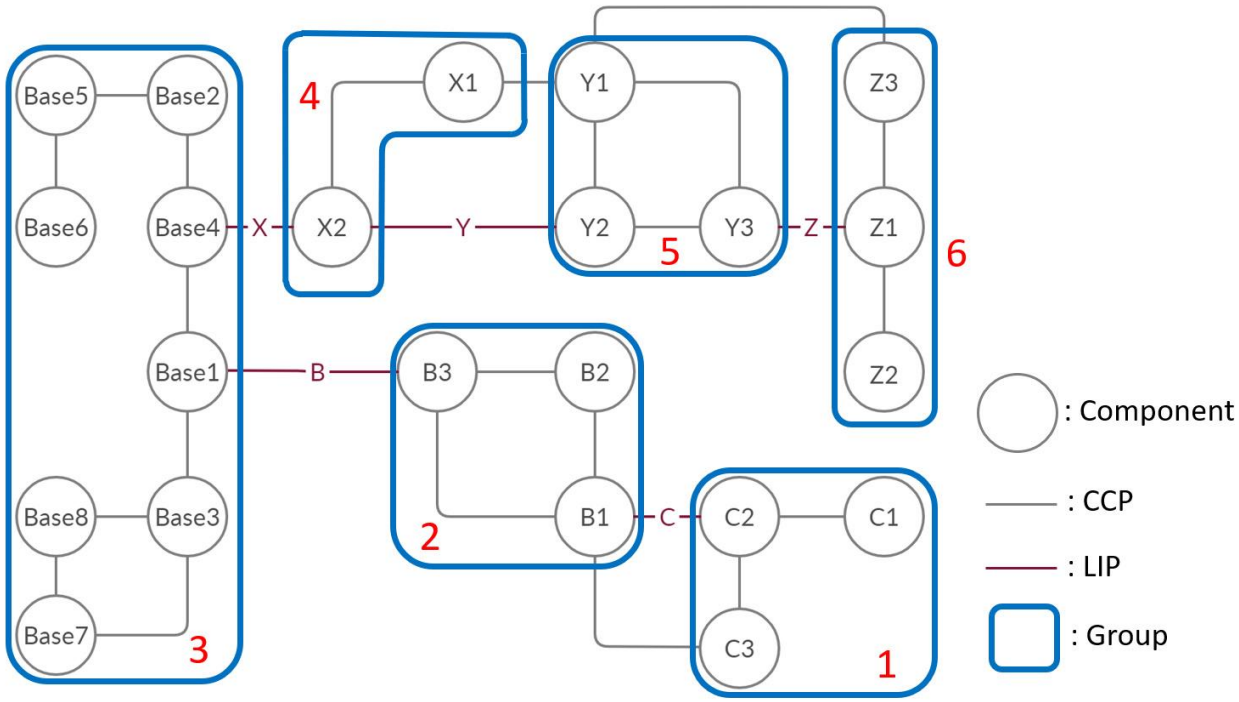

Figure 5.4 Final result of link-component grouping of UMC-500 example

It is possible that link collision still exists after trying all possible grouping for components that cause link collision and uncertainties when the voxel size is too big relative to the components. Since link collision is detected by using the offset voxel models. If the voxel size is relatively too big, the triangular mesh models might have empty triangular mesh models after offsetting because the offset distance is larger than the components. So, the original triangular mesh models are used to get voxel models for validation of link groups. Because voxel size is relatively big and the original triangular mesh models are voxelized to get voxel models, false collision can happen during the validation of link groups. False collision causes link collision to keep existing no matter how the resolution of link collision regroups the components to generate the new link groups and it fails to generate link groups to export into the kinematic chain exporter.

Chapter 6: Implementation Results

Five commercial Haas five-axis machine tools with varying configurations have been implemented using the presented method. Haas is the largest machine tool builder in the western world. They build a complete line of CNC vertical machining centers, horizontal machining centers, and CNC lathes. Haas provides assembly files of certain products on the website [10] for customers working on shop layout or figuring out how CAD-modeled parts will fit into their machines. Eight assembly files of five-axis machine tools are provided on Haas website with the travel span of each axis for each machine. Five of these machine tools were used for this work because the other three assembly files of machine tools have obvious missing components. The implementations were performed on a PC equipped with a 64-bit Intel i7 CPU (2.9 GHz) and 8 GB of RAM.

Among the five Haas five-axis machine tool, VF-2TR has a trunnion that is mounted directly to the machine's table to provide additional rotary axes in A and C axes. UMC-500 is a five-axis machine tool that has two rotary axes in B and C axes. UMC-750 is a larger version of UMC-500 with longer travel spans in X, Y, and Z axis. UMC-1600H is a five-axis horizontal mill with rotary axes in A and C axes. These four five-axis machine tools are 3-2 configuration type and table-rotating structural type. VR-8 is a five-axis machine tool with a dual-axis spindle head. It has two rotary axes in A and C axes and is 3-2 configuration type and spindle-rotating structural type.

Session 6.1 demonstrates the results that use single-level voxel size for the link-component grouping and the validation of link groups for these machine tools. In order to increase the efficiency of the whole process, two-level voxel size that uses different voxel sizes for the link-

component grouping and the validation of link groups is implemented, and the results are shown in session 6.2.

6.1 Single-Level Voxel Size

Single-level voxel size indicates that the link-component grouping and the validation of link groups use the same voxel size Table 6-1 to Table 6-10 show the computational times of using five different voxel sizes with two different strategies for the five machines. Each movable link has 5 sampled poses for the validation of link groups. When an uncertainty exists, the validation of link groups is mandatory. "-" in the tables indicates no resolution is applied to the corresponding type of uncertainty. The kinematic chains in URDF and STL files of each link of machines generated by the presented method are input to ROS. ROS is used because it can read URDF, visualize links according to the URDF, and move each link within its travel span manually. To demonstrate the movements of axes of machines within travel spans with the generated kinematic chains and STL files, the neural pose and the poses after moving each moveable link of four machine tools with the coordinate frame of each link in ROS environment are shown in Figure 6.1 to Figure 6.4. As enclosures block the view to see links of machine tools move, they are excluded for the demonstration purpose in this session. UMC-750 is not demonstrated since it is similar to UMC-500 but with longer travel spans.

Table 6-1 Results of VF-2TR using strategy A (Single-level voxel size)

VF-2TR (Strategy A)	2mm voxel size		3mm voxel size		4mm voxel size		5mm voxel size		6mm voxel size	
Contains uncertainty?	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2
	No	Yes	Yes	Yes	Yes	Yes	Failed		Failed	
Validation of link groups	Necessary		Necessary		Necessary		Failed		Failed	
Number of resolution	-	0	1	0	1	0	Failed		Failed	
Computational Time [minute]	405.6		79.7		47.4		Failed		Failed	

Table 6-2 Results of VF-2TR using strategy B (Single-level voxel size)

VF-2TR (Strategy B)	2mm voxel size		3mm voxel size		4mm voxel size		5mm voxel size		6mm voxel size	
Contains uncertainty?	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2
	No	Yes	Yes	Yes	Yes	Yes	Failed		Failed	
Validation of link groups	Necessary		Necessary		Necessary		Failed		Failed	
Number of resolution	-	0	0	0	0	0	Failed		Failed	
Computational Time [minute]	404.8		51.9		29.9		Failed		Failed	

Table 6-3 Results of UMC-500 using strategy A (Single-level voxel size)

UMC-500 (Strategy A)	2mm voxel size		3mm voxel size		4mm voxel size		5mm voxel size		6mm voxel size	
Contains uncertainty?	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2
	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Failed	
Validation of link groups	Necessary		Necessary		Necessary		Necessary		Failed	
Number of resolution	0	0	0	0	0	0	1	0	Failed	
Computational Time [minute]	290.1		69.3		21.7		30.5		Failed	

Table 6-4 Results of UMC-500 using strategy B (Single-level voxel size)

UMC-500 (Strategy B)	2mm voxel size		3mm voxel size		4mm voxel size		5mm voxel size		6mm voxel size	
Contains uncertainty?	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2
	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Failed	
Validation of link groups	Necessary		Necessary		Necessary		Necessary		Failed	
Number of resolution	1	0	2	0	2	0	2	0	Failed	
Computational Time [minute]	542.3		187.0		57.6		43.4		Failed	

Table 6-5 Results of UMC-750 using strategy A (Single-level voxel size)

UMC-750 (Strategy A)	2mm voxel size		3mm voxel size		4mm voxel size		5mm voxel size		6mm voxel size	
Contains uncertainty?	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2
	No	No	No	No	No	No	No	No	Failed	
Validation of link groups	Unnecessary		Unnecessary		Unnecessary		Unnecessary		Failed	
Number of resolution	-	-	-	-	-	-	-	-	Failed	
Computational Time [minute]	59.6		27.3		6.9		2.2		Failed	

Table 6-6 Results of UMC-750 using strategy B (Single-level voxel size)

UMC-750 (Strategy B)	2mm voxel size		3mm voxel size		4mm voxel size		5mm voxel size		6mm voxel size	
Contains uncertainty?	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2
	No	No	No	No	No	No	No	No	Failed	
Validation of link groups	Unnecessary		Unnecessary		Unnecessary		Unnecessary		Failed	
Number of resolution	-	-	-	-	-	-	-	-	Failed	
Computational Time [minute]	59.2		27.6		6.7		2.3		Failed	

Table 6-7 Results of UMC-1600H using strategy A (Single-level voxel size)

UMC-1600H (Strategy A)	2mm voxel size		3mm voxel size		4mm voxel size		5mm voxel size		6mm voxel size	
Contains uncertainty?	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2
	No	No	No	Yes	Yes	Yes	Failed		Failed	
Validation of link groups	Unnecessary		Necessary		Necessary		Failed		Failed	
Number of resolution	-	-	-	0	1	0	Failed		Failed	
Computational Time [minute]	57.8		677.4		176.7		Failed		Failed	

Table 6-8 Results of UMC-1600H using strategy B (Single-level voxel size)

UMC-1600H (Strategy B)	2mm voxel size		3mm voxel size		4mm voxel size		5mm voxel size		6mm voxel size	
Contains uncertainty?	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2
	No	No	No	Yes	Yes	Yes	Failed		Failed	
Validation of link groups	Unnecessary		Necessary		Necessary		Failed		Failed	
Number of resolution	-	-	-	0	0	0	Failed		Failed	
Computational Time [minute]	57.5		676.8		92.5		Failed		Failed	

Table 6-9 Results of VR-8 using strategy A (Single-level voxel size)

VR-8 (Strategy A)	2mm voxel size		3mm voxel size		4mm voxel size		5mm voxel size		6mm voxel size	
Contains uncertainty?	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2
	No	No	No	No	No	No	No	No	Failed	
Validation of link groups	Unnecessary		Unnecessary		Unnecessary		Unnecessary		Failed	
Number of resolution	-	-	-	-	-	-	-	-	Failed	
Computational Time [minute]	54.8		23.4		8.2		6.2		Failed	

Table 6-10 Results of VR-8 using strategy B (Single-level voxel size)

VR-8 (Strategy B)	2mm voxel size		3mm voxel size		4mm voxel size		5mm voxel size		6mm voxel size	
Contains uncertainty?	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2	Type1	Type2
	No	No	No	No	No	No	No	No	Failed	
Validation of link groups	Unnecessary		Unnecessary		Unnecessary		Unnecessary		Failed	
Number of resolution	-	-	-	-	-	-	-	-	Failed	
Computational Time [minute]	54.2		23.3		8.4		6.1		Failed	

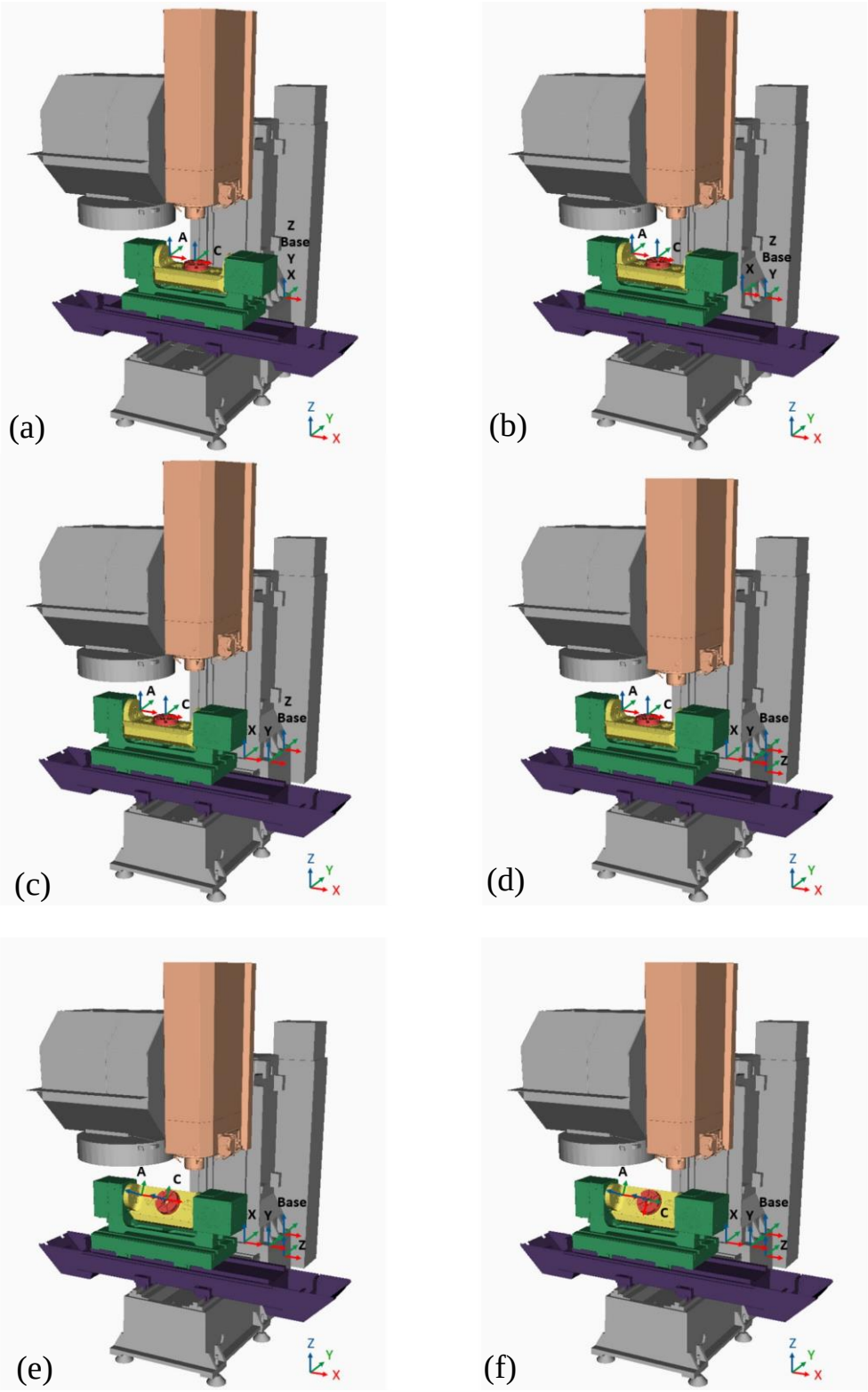

Figure 6.1 Kinematic chain demonstration of VF-2

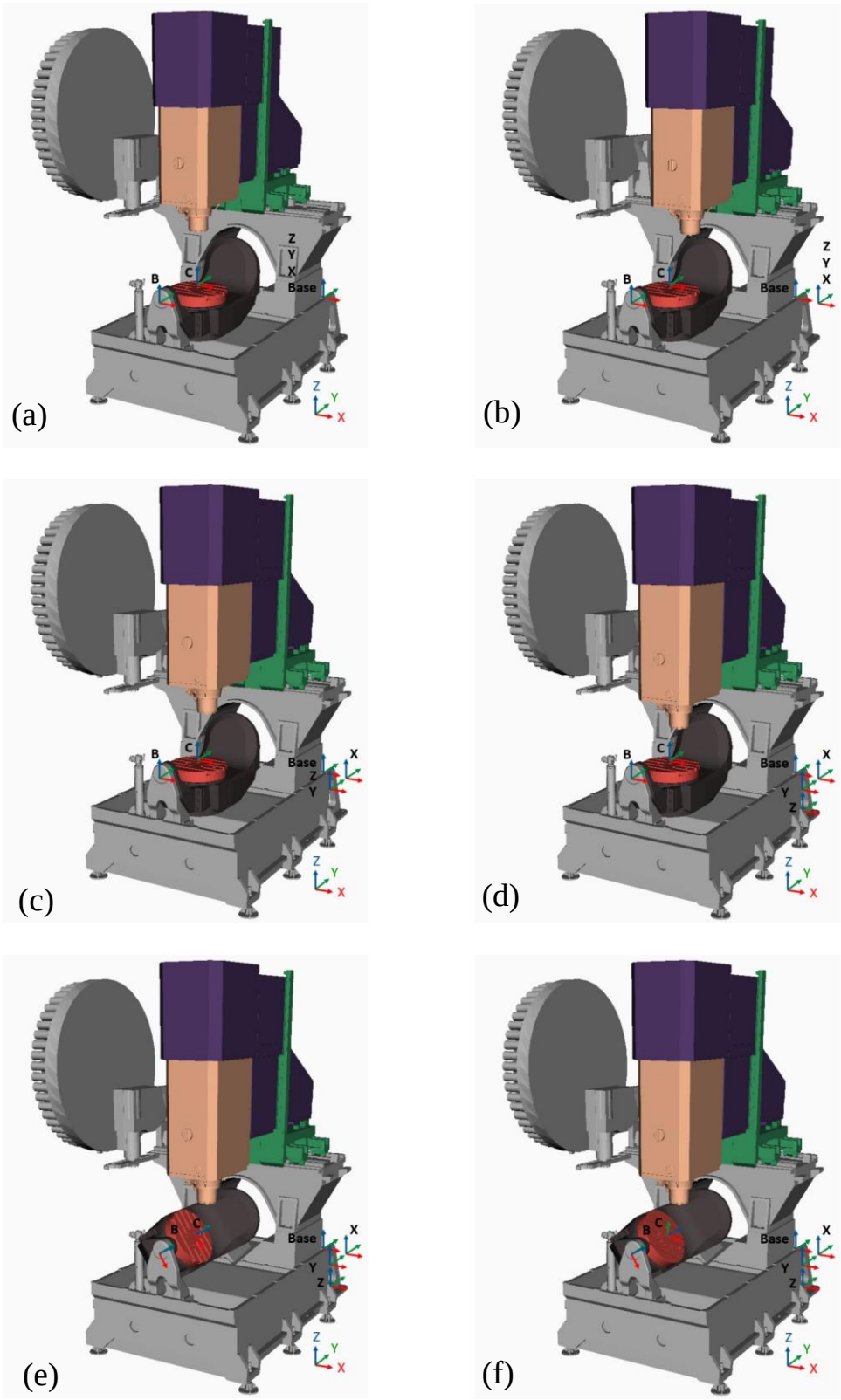

Figure 6.2 Kinematic chain demonstration of UMC-500

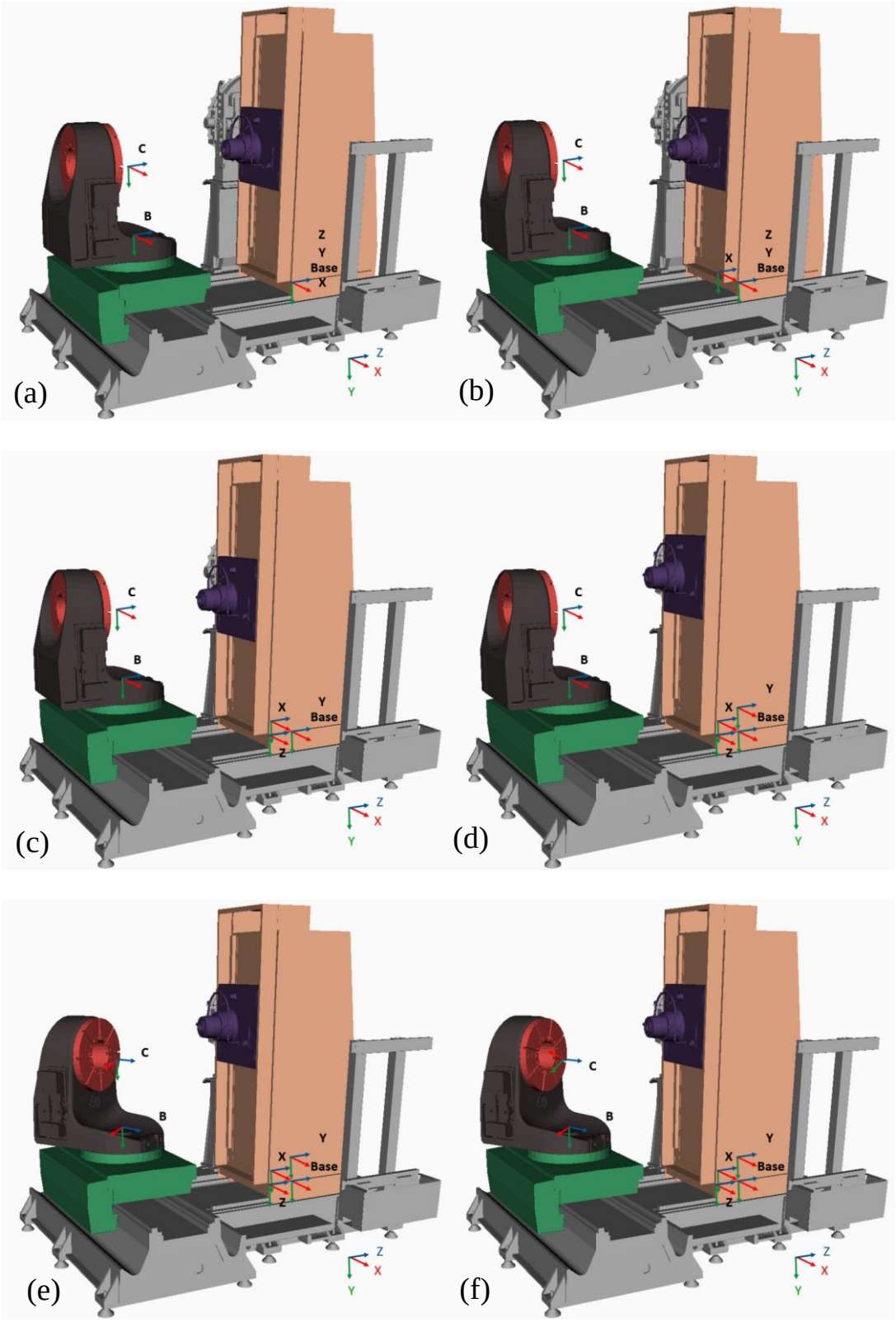

Figure 6.3 Kinematic chain demonstration of UMC-1600H

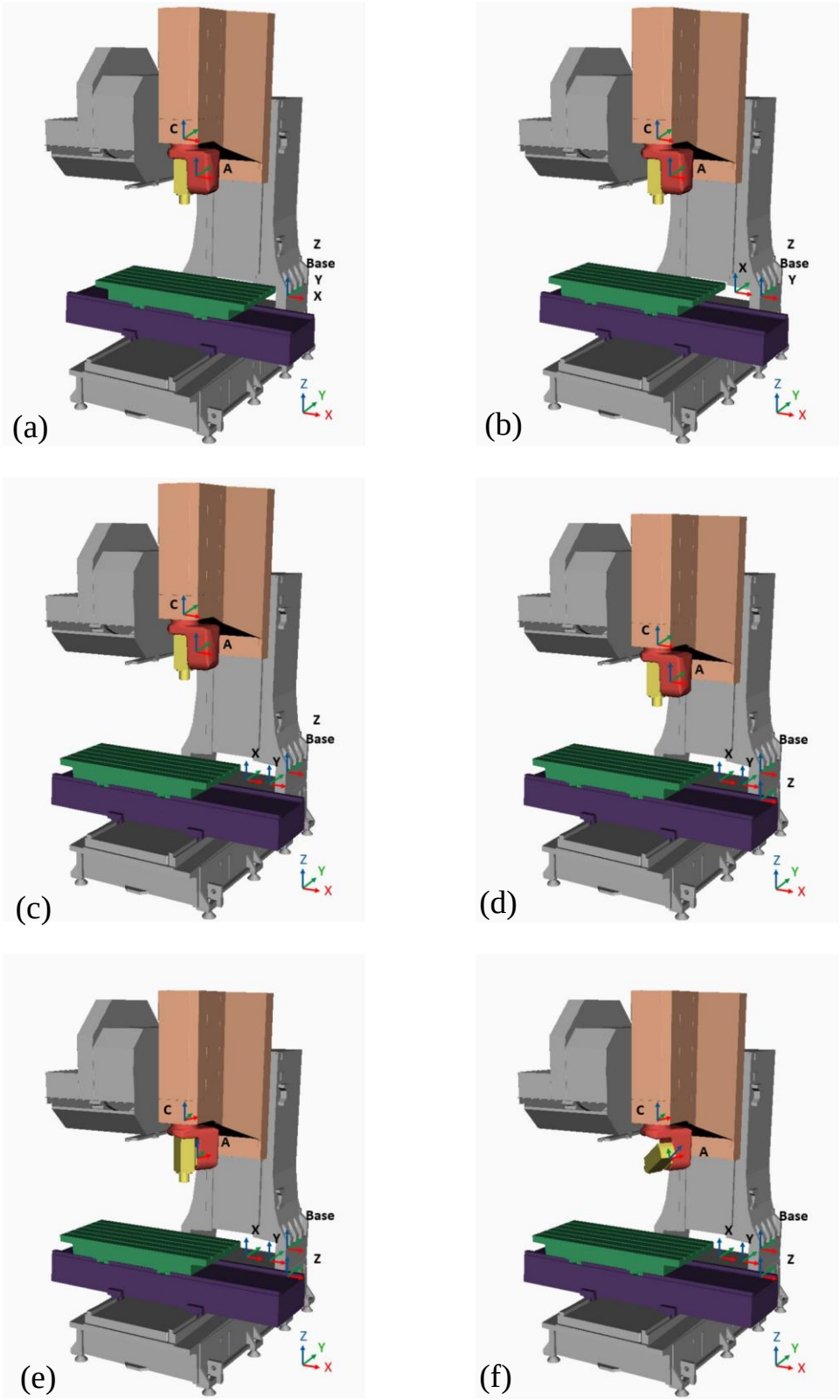

Figure 6.4 Kinematic chain demonstration of VR-8

From the results of these five five-axis machine tools above using 5 different voxel sizes and two different strategies, the presented automatic grouping method that uses single-level voxel size with basic user inputs can successfully group components for links of machine tools and generate correct kinematic chains. The correct kinematic chains and STL file of each link for these machine tools have been generated and can be used for simulation of machine movements.

It can be observed that using a smaller voxel size leads to fewer uncertainty. Also, if an uncertainty exists, using a smaller voxel size requires a fewer number of resolutions. It is because a smaller voxel size causes fewer false CCPs created by false interference. With fewer false CCPs, it has a lower chance to have an uncertainty from link-component grouping and a lower chance to generate incorrect link groups that cause link collision. However, due to the nature of a smaller voxel size, using a smaller voxel size requires more memory and computational time. 2 mm is the smallest voxel size used in this work because it is the smallest voxel size within the computer memory limit to implement the presented method on the five five-axis machine tools.

While a larger voxel size is used, it causes the presented method to fail to generate the desired results. It is because contact features that can be used to identify LIPs from CCPs could disappear after models are offset with a larger offset distance and it leads to the number of LIPs is fewer than the number of axis of the machine tool. Further steps of the method are terminated and the method fails to generate the kinematic chain and STL files of the machine. Therefore, despite a shorter computational time is taken by using a larger voxel size, the voxel size cannot be too big, otherwise, the presented method could fail.

It is important to note that when a smaller voxel size is used, most of the computing time is taken by the validation of link groups if it is necessary, the link-component grouping only takes a small portion of the total computing time. Using a smaller voxel size is preferred for the link-component grouping since it has a lower chance of having an uncertainty leading to the necessity for the validation of link groups. Also, it generates less false CCPs that could require a greater number of the resolution of link collision. On the other hand, the grouping validation doesn't require a smaller voxel size. It can use a bigger voxel size to increase the efficiency of the whole process to generate kinematic chai of machines with less computing time. Thus, a method using two-level voxel size is presented in the next session.

6.2 Two-Level Voxel Size

Since a smaller voxel size is better for link-component grouping to have fewer false CCPs that can cause incorrect link groups, two-level voxel size is presented to use 2 mm voxel size for the link-component grouping and 3 different larger voxel sizes for the validation of link groups. Each movable link has 5 sampled poses for the validation of link groups. The results of using two-level voxel size are shown in Table 6-11 to Table 6-14. Only UMC-500 and VF-2TR using two-level voxel size are shown as there is no uncertainty from other machines using 2mm voxel size for link-component grouping. No uncertainty indicates it is unnecessary to have the validation of link groups and resolution of link collision. Therefore, two-level voxel size cannot improve efficiency and have similar results of using single-level voxel size for the other four machines.

Table 6-11 Results of VF-2TR using strategy A (two-level voxel size)

VF-2TR (Strategy A)	5 mm voxel size		6 mm voxel size		8 mm voxel size	
Contains uncertainty?	Type1	Type2	Type1	Type2	Type1	Type2
	No	Yes	No	Yes	No	Yes
Validation of link groups	Necessary		Necessary		Necessary	
Number of resolution	-	0	-	0	-	0
Computational Time [minute]	31.6		27.7		25.4	

Table 6-12 Results of VF-2TR using strategy B (two-level voxel size)

VF-2TR (Strategy B)	5 mm voxel size		6 mm voxel size		8 mm voxel size	
Contains uncertainty?	Type1	Type2	Type1	Type2	Type1	Type2
	No	Yes	No	Yes	No	Yes
Validation of link groups	Necessary		Necessary		Necessary	
Number of resolution	-	0	-	0	-	0
Computational Time [minute]	31.2		27.8		25.1	

Table 6-13 Results of UMC-500 using strategy A (two-level voxel size)

UMC-500 (Strategy A)	5 mm voxel size		6 mm voxel size		8 mm voxel size	
Contains uncertainty?	Type1	Type2	Type1	Type2	Type1	Type2
	Yes	Yes	Yes	Yes	Yes	Yes
Validation of link groups	Necessary		Necessary		Necessary	
Number of resolution	0	0	0	0	Failed	
Computational Time [minute]	43.1		40.5		Failed	

Table 6-14 Results of UMC-500 using strategy B (two-level voxel size)

UMC-500 (Strategy B)	5 mm voxel size		6 mm voxel size		8 mm voxel size	
Contains uncertainty?	Type1	Type2	Type1	Type2	Type1	Type2
	Yes	Yes	Yes	Yes	Yes	Yes
Validation of link groups	Necessary		Necessary		Necessary	
Number of resolution	1	0	1	0	Failed	
Computational Time [minute]	57.8		50.6		Failed	

From the results above, using two-level voxel size reduces computational time significantly to generate correct kinematic chains and group link components for machines compared to single-level voxel. Despite that using a bigger voxel size for the validation of link groups can decrease computational time, a bigger voxel size can cause mesh models of components to disappear after offsetting because a bigger voxel size indicates a larger offset distance. The original mesh models then need to be used for voxelization and interference detection. The validation of link-groups using voxel models voxelized from original mesh models can lead to false link collision. As a result, the method cannot generate the kinematic chain as the false link collision cannot be solved by the resolution of link collision. 8mm voxel size in the tables are the cases that the method fails to generate kinematic chains because false link collision exists during the validation of link groups and it cannot be resolved.

Chapter 7: Conclusions and Future Works

7.1 Conclusions

In this work, an automatic method to group link components to generate kinematic chains and the STL file of each link of multi-axis machine tools for motion simulation has been presented. The method requires only the assembly file, axes and travel span of each link of the given machine tool to avoid an error-prone and lengthy manual process. Since the uncertainty of link-component grouping could lead to incorrect link groups, the validation of link groups is necessary when an uncertainty exists. If no link collision caused by incorrect link groups occurs, the generated link groups can be exported directly to the kinematic chain exporter to generate the kinematic chain and STL file of each link of the given machine. Otherwise, the resolution of link collision is used to regroup the component that leads to an uncertainty and link collision. The new link groups need to go through the iterative step of validation and resolution until no link collision exists.

The presented method has been implemented on five commercial Haas five-axis machine tools using single-level voxel size. The results of kinematic chains and computational times for different voxel sizes showed that the ability to generate correct kinematic chains and STL files with only basic inputs for simulation of machine movements. Because most of the computational time is taken by the link-components grouping which requires a smaller voxel size and the validation of link groups doesn’t require a smaller voxel size, two-level voxel size was presented to use a smaller voxel size for the link-component grouping and a bigger voxel sizer for the validation of link groups. The result showed that the computational time is significantly reduced by using two-level voxel size to generate the same kinematic chains as single-level voxel size.

7.2 Future Works

An improvement to the work presented in this thesis is to develop a more efficient voxelization method for the link-component grouping. Although the current method using two-level voxel size exhibited a reasonable degree of efficiency, in order to reduce the chance of causing the uncertainty from link-component grouping, the method requires using voxel size as smaller as possible which increases computational time and burden of computer memory to store voxel information. Developing a voxelization method for link-component grouping which is more efficient than the current method could improve the accuracy of interference detection and a lower chance to generate uncertainties and incorrect link groups.

The automatic grouping method presented in this thesis can generate kinematic chains and STL files only when the number of LIPs is equal to the number of axis of the given machine tool and only one LIP can exist between two link groups. If this work considers the scenario of the number of LIPs is larger than the number of axis of the given machine tool and there can be more than one LIP between two link groups, it would require a more comprehensive link-component grouping and the strategy to handle the uncertainty from grouping.

Bibliography

[1] R. V. Fleisig and A. D. Spence, "A constant feed and reduced angular acceleration interpolation algorithm for multi-axis machining," *Computer-Aided Design,* vol. 33, no. 1, pp. 1-15, 2001.

[2] D. Jang, K. Kim, and J. Jung, "Voxel-based virtual multi-axis machining," *The International Journal of Advanced Manufacturing Technology,* vol. 16, no. 10, pp. 709-713, 2000.

[3] S. Ding, M. Mannan, and A. N. Poo, "Oriented bounding box and octree based global interference detection in 5-axis machining of free-form surfaces," *Computer-Aided Design,* vol. 36, no. 13, pp. 1281-1294, 2004.

[4] O. Ilushin, G. Elber, D. Halperin, R. Wein, and M.-S. Kim, "Precise global collision detection in multi-axis NC-machining," *Computer-Aided Design,* vol. 37, no. 9, pp. 909-920, 2005.

[5] T. Tang, E. L. Bohez, and P. Koomsap, "The sweep plane algorithm for global collision detection with workpiece geometry update for five-axis NC machining," *Computer-Aided Design,* vol. 39, no. 11, pp. 1012-1024, 2007.

[6] B. Lauwers, P. Dejonghe, and J.-P. Kruth, "Optimal and collision free tool posture in five-axis machining through the tight integration of tool path generation and machine simulation," *Computer-Aided Design,* vol. 35, no. 5, pp. 421-432, 2003.

[7] Y. Altintas, C. Brecher, M. Weck, and S. Witt, "Virtual machine tool," *CIRP annals,* vol. 54, no. 2, pp. 115-138, 2005.

[8] "Robot Operating System (ROS)," [Online]. Available:https://www.ros.org/.

[9] "SolidWorks URDF Exporter," [Online]. Available:http://wiki.ros.org/sw_urdf_exporter.

[10] "Haas Automation Inc.," [Online]. Available:https://www.haascnc.com/index.html.

[11] E. L. Bohez, "Five-axis milling machine tool kinematic chain design and analysis," *International Journal of Machine Tools and Manufacture,* vol. 42, no. 4, pp. 505-520, 2002.

[12] S. Sakamoto, "Analysis of generating motion for five-axis machining centers," *Transactions of NAMRI/SME,* p. 287, 1993.

[13] R.-S. Lee and K.-J. Mei, "Motion and virtual cutting simulation system for a five-axis virtual machine tool," *International Journal of Automation and Smart Technology,* vol. 1, no. 1, pp. 35-39, 2011.

[14] F. Chen, "On the structural configuration synthesis and geometry of machining centres," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science,* vol. 215, no. 6, pp. 641-652, 2001.

[15] Y. Lin and Y. Shen, "Modelling of five-axis machine tool metrology models using the matrix summation approach," *The International Journal of Advanced Manufacturing Technology,* vol. 21, no. 4, pp. 243-248, 2003.

[16] R.-S. Lee and C.-H. She, "Developing a postprocessor for three types of five-axis machine tools," *The International Journal of Advanced Manufacturing Technology,* vol. 13, no. 9, pp. 658-665, 1997.

[17] M. Tsutsumi and A. Saito, "Identification of angular and positional deviations inherent to 5-axis machining centers with a tilting-rotary table by simultaneous four-axis control movements," *International Journal of Machine Tools and Manufacture,* vol. 44, no. 12-13, pp. 1333-1342, 2004.

[18] O. R. Tutunea-Fatan and H.-Y. Feng, "Configuration analysis of five-axis machine tools using a generic kinematic model," *International Journal of Machine Tools and Manufacture*, vol. 44, no. 11, pp. 1235-1243, 2004.

[19] R. S. Hartenberg and J. Denavit, "A kinematic notation for lower pair mechanisms based on matrices," *Journal of applied mechanics*, vol. 77, no. 2, pp. 215-221, 1955.

[20] A. Yousuf, W. Lehman, M. A. Mustafa, and M. M. Hayder, "Introducing kinematics with robot operating system (ROS)," in *122nd ASEE Annual Conference and Exposition*, 2015, pp. 1-18.

[21] L. Kunze, T. Roehm, and M. Beetz, "Towards semantic robot description languages," in *2011 IEEE International Conference on Robotics and Automation*, 2011: IEEE, pp. 5589-5595.

[22] "XML Robot Description Format (URDF)," [Online]. Available:http://wiki.ros.org/urdf/XML/model.

[23] Y. Kang, D. Kim, and K. Kim, "URDF Generator for Manipulator Robot," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, 2019: IEEE, pp. 483-487.

[24] "SolidWorks," [Online]. Available:https://www.solidworks.com/.

[25] H. Deng, J. Xiong, and Z. Xia, "Mobile manipulation task simulation using ROS with Moveit," in *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2017: IEEE, pp. 612-616.

[26] S. Thongnuch and A. Fay, "A practical simulation model generation for virtual commissioning," in *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2017: IEEE, pp. 1077-1082.

[27] P. Vyavahare, S. Jayaprakash, and K. Bharatia, "Construction of URDF model based on open source robot dog using Gazebo and ROS," in *2019 Advances in Science and Engineering Technology International Conferences (ASET)*, 2019: IEEE, pp. 1-5.

[28] M. Moore and J. Wilhelms, "Collision detection and response for computer animation," in *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, 1988, pp. 289-298.

[29] A. Kaufman, D. Cohen, and R. Yagel, "Volume graphics," *Computer*, vol. 26, no. 7, pp. 51-64, 1993.

[30] A. E. Kaufman, "Introduction to volume graphics," in *SIGGRAPH*, 1999, vol. 99, pp. 24-47.

[31] E. Özbay and A. Çinar, "A voxelize structured refinement method for registration of point clouds from Kinect sensors," *Engineering Science and Technology, an International Journal*, vol. 22, no. 2, pp. 555-568, 2019.

[32] M. Sagardia, T. Hulin, C. Preusche, and G. Hirzinger, "Improvements of the voxmap-pointshell algorithm-fast generation of haptic data-structures," in *53rd IWK-Internationales Wissenschaftliches Kolloquium, Ilmenau, Germany*, 2008.

[33] T. Akenine-Möllser, "Fast 3D triangle-box overlap testing," *Journal of graphics tools*, vol. 6, no. 1, pp. 29-33, 2001.

[34] S. Lock and D. P. Wills, "VoxColliDe: Voxel collision detection for virtual environments," *Virtual Reality*, vol. 5, no. 1, pp. 8-22, 2000.

[35] W. Lin and J. Fu, "Modeling and application of virtual machine tool," in *16th International Conference on Artificial Reality and Telexistence--Workshops (ICAT'06)*, 2006: IEEE, pp. 16-19.

[36] S.-H. Suh, Y. Seo, S.-M. Lee, T.-H. Choi, G.-S. Jeong, and D.-Y. Kim, "Modelling and implementation of internet-based virtual machine tools," *The International Journal of Advanced Manufacturing Technology,* vol. 21, no. 7, pp. 516-522, 2003.

[37] K.-J. Mei and R.-S. Lee, "Collision detection for virtual machine tools and virtual robot arms using the Shared Triangles Extended Octrees method," *International Journal of Computer Integrated Manufacturing,* vol. 29, no. 4, pp. 355-373, 2016.

[38] C.-H. She and C.-C. Chang, "Design of a generic five-axis postprocessor based on generalized kinematics model of machine tool," *International Journal of Machine Tools and Manufacture,* vol. 47, no. 3-4, pp. 537-545, 2007.

[39] R. S. Lee and Y. H. Lin, "Development of universal environment for constructing 5-axis virtual machine tool based on modified D–H notation and OpenGL," *Robotics and Computer-Integrated Manufacturing,* vol. 26, no. 3, pp. 253-262, 2010.

[40] J. Yang and Y. Altintas, "Generalized kinematics of five-axis serial machines with non-singular tool path generation," *International Journal of Machine Tools and Manufacture,* vol. 75, pp. 119-132, 2013.

[41] "Autodesk Meshmixer," [Online]. Available:https://www.meshmixer.com/.

[42] "OpenCascade," [Online]. Available:https://www.opencascade.com/.