UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Génie Industriel**

Arrêté ministériel : 7 août 2006

Présentée par

Chenggang WANG

Thèse dirigée par **Peter MITROUCHEV**

Préparée au sein du **Laboratoire G-SCOP**
Dans l'**Ecole doctorale I-MEP2 - Grenoble INP**

Génération des séquences de désassemblage et leur évaluation. Intégration dans un environnement de réalité virtuelle

Thèse soutenue publiquement le 06.11.2014
devant le jury composé de :

M. Marc SARTOR	
Professeur INSA Toulouse,	Président
M. Georges DUMONT	
Professeur ENS de Rennes,	Rapporteur
M. Grigore GOGU	
Professeur IFMA Clermont-Ferrand,	Rapporteur
M. Michel TOLLENAERE	
Professeur Grenoble INP,	Membre
M. Lixin LU	
Professeur Université de Shanghai,	Co-directeur de thèse
M. Peter MITROUCHEV	
MCF-HC, Université de Grenoble,	Directeur de thèse

Université Joseph Fourier / Université Pierre Mendès France /
Université Stendhal / Université de Savoie / Grenoble INP

Dedication

To my parents,

my wife and my newborn daughter

Acknowledgment

Foremost, I would like to thank my advisor Dr. Peter Mitrouchev for the continuous support of my PhD study and research, for his patience, motivation, enthusiasm and immense knowledge. I would like to express my sincere gratitude to Grenoble INP for letting me fulfill my dream of being a PHD student here. I would like to thanks the staff of G-SCOP Laboratory for their good reception.

I wish to thank all members of the jury for the attention they have kindly given to this work: Prof. SARTOR Marc for doing me honors to chair this jury, Prof. DUMONT Georges and Prof. GOGU Grigore for accepting the burden to be Reviewers and Prof. TOLLENAERE Michel for allowing me the honor to consider it. They find here my gratitude for having taken some of their time to read this brief and give constructive criticism.

My sincere thanks also goes to Prof. Frédéric NOEL from Grenoble INP and his Visionair team, for supporting me with the device for simulation, the valuable advice for the development of the application, the software, and also for supporting the two international conferences fees.

I would like to thank Dr. Guiqin LI, from Shanghai University, who provide an opportunity for study here. Many thanks give to Prof. Lixin LU, from Shanghai University, who is always willing to help when I need to.

I would like to thank the French Ministry of High Education for awarding me a scholarship and providing me with the facilities to complete this thesis.

I would like to thank all the PHD students of GSCOP Laboratory for the exchanges of ideas and advices during the last three years.

List of the figures

List of the tables

List of the articles and the communications

I. Related with the PHD thesis

International Journals:

1. Chenggang Wang, Peter Mitrouchev, Guiqin LI. Disassembly operations' efficiency evaluation in virtual environment, International Journal of Computer Integrated Manufacturing (under press).

2 Chenggang Wang, Peter Mitrouchev, Guiqin LI. Sequences planning for products disassembly based on lowest levels disassembly graph method. International journal of advanced manufacturing technology (22 pages, under review).

International Conferences:

1. Chenggang Wang, Peter Mitrcouchev. 3D Geometric Removability Analysis for Virtual Disassembly Evaluation. AIM 2014, IEEE/ASME International Conference on Advanced Intelligent Mechatronics. July 8-11, 2014 Besançon, France. 6 pages (CD-ROM, paper N°DETC2014-34943).

2. Chenggang Wang, Peter Mitrouchev. Least Levels disassembly graph method for selective disassembly planning. Proceedings of ASME 2014 International Design and Engineering Technical Conferences& Computers and Information in Engineering Conference. IDETC/CIE 2014 August 17-20, 2014 in Buffalo, NY, USA. 10 pages (CD-ROM, paper N°#264).

II. Others

International Journals:

1. Zhang S., Li G., Lu L., Mitrouchev P., Wang C.-G. Numerical Analysis of Multi-layer Continuous Diffusion Furnace Door Gas Curtain. Applied Mechanics and Materials Vol. 367 (2013). pp 462-465.

2. Li Qui-qin, Mitrouchev P., Wang C.-G., Brissaud D., Lu L. Evaluation of the logistics model of Reconfigurable Manufacturing System based on Generalized Stochastic Petri Nets. International Journal of Production Research, Volume 50, Issue 22, 2012, (TPRS-2011-IJPR-0529), Ed. Taylor & Francis, ISSN 0020–7543 print/ISSN 1366–588X online.

National Journals:

1. Wang Cheng-gang, LI Gui-qin, Lu Li-in, Wang Yan. Uniform airflow Research on Plasma Enhanced Chemical Vapor Deposition [J]. Computer simulation. ISSN 1006-9348/CN11-3724/TP.

2. Jin Jie, Zhu Hongping, Wang Chenggang. Airflow field simulation analysis of PEVCD gas distribution device. Modern Manufacturing Engineering. 12(2), 2012.

3. LI Gui-qin, LU Li-in, Wang Cheng-gang. Numerical simulation of the flow field in PECVD reaction chamber, Vacuum, 49(3), 2012.

Table of contents

General introduction

A few decades ago, the products were designed mainly to answer the customer requirements and the possibilities of the manufacturers without taking into account the environmental aspects and those of recycling during the design process. However, with the implementation of new European and International standards and recommendations of environmental legislation, the problem of dismantling and thereafter of products recycling is increasingly important. The economical and environmental consequences of products disassembling and re-using at the end of their life cycle are also to be taken into account. Thus, today's designers need new tools allowing them in particular generating and evaluating disassembly operations.

Preservation of the environment and the planet's resources is currently being a great concern. Awareness of the heavy environmental impact of production has led to a new field of research concerning the recycling of the End–Of–Life (EOL) products. Two main methods are used in this field: *shredding* and *disassembly*. Shredding is a quick way of recycling materials but its main drawback is the impurity of the recovered (produced) material. Thus, in order to decrease the material's impurity and to reclaim higher value components, effective methods of dissembling the products appear very important. Nowadays and in a near future, the demand for high productivity and increasing labor cost are pushing designers to improve the effectiveness of disassembly processes.

From the disassembly point of view, there are two main types of disassembly methods. One is *complete* disassembly which involves disassembling of all the components of an assembly. However, it is rarely the optimal solution due to the high costs of the disassembly process. Alternatively, *selective* disassembly is usually more appropriate for manufacturing applications, such as: maintenance, repairing or recycling. Currently, there are two types of disassembly environments for generating the disassembly sequences: *interactive* and *automated*. However, both of them have some limitations. Interactive environments, for instance, require extensive user input usually in the form of answering questions, whereas automated ones can only be used to generate disassembly processes for products with relatively simple component configuration and geometry because of the tremendous amount of computation resources required. The mostly used methods for disassembly sequence generation are *heuristic searching algorithms*, *linear programming*, *wave propagation*.

Recently new Artificial Intelligence (AI) based methods such as: *Expert Systems*, *Petris Nets*, *Genetic Algorithms* are proposed as well. Almost all these methods are used in special situations or need computation resources for complex products. Thus we need a general and complete model, able to describe the allowed movements for components during the simulations of disassembly operations of *interactive*, *real-time* or *immersive* types.

Often, after generating the possible disassembly sequences, it is necessary to evaluate them. Virtual prototyping is quickly becoming an interesting strategy for the product development and Assembly/Disassembly (A/D) operations evaluation in recent years. Almost in all fields related to the product development process (PDP), virtual simulation using virtual representation models of the products are created in a virtual environment (VE). Thus, virtual prototyping (VP) is quickly becoming a strategy in the PDP. It allows understanding the application of virtual reality (VR) for the prototyping physical mock-up by using product and process data. Simulations closely related with VR environments represent important research subject. A major role is played by assembly/disassembly (A/D) operations in the initial stages of the product design, such as: production, ergonomics, training, health, service or recycling stages. VR technology plays a vital role in simulating such advanced 3D human-computer interaction by providing users with different kinds of sensations such as: visual, auditory, haptic. Virtual assembly simulations allow designers to evaluate the concepts in virtual environments during the early design stage. With virtual prototyping applications, optimizing the design for assembly process can be incorporated easily in the conceptual design stage. Using haptics or auditory technology allows designers to interact with the parts with the human basic motions. Thus, contact force for instance may be transmitted to the operator in real time.

In recent years Ergonomic assessment of manufacturing industry in VR system is becoming increasingly globalized as well. The purpose of the Ergonomic assessment is to try to fit the task to the human and not the human to the task. The key point for an effective application is to gain a balance between the human body characters and the task demands.

Research problematic

As mentioned here above, integration of disassembly operations during product design is an important issue today. As known, the number of possible disassembly sequences significantly increases with the number of parts in a product. Thus, the generation of proper disassembly sequences order is critical. Most of existing methods often require tremendous computational resources while, at the same time, they often fail to find realistic and optimal solutions for complex products disassembly.

Disassembly operations cover a broad range of the Product Life Cycle (PLC) regarding operations of disassembly during: production process, product maintenance and finally at the end of PLC. It is

estimated that at the earliest stages of product design, the cost of these operations almost represents 30% of its total cost. Modelling these operations requires a lot of geometrical, kinematical, technological and ergonomical data and their synthesis in order to reduce the algorithmic complexity of the disassembly simulation process. Nowadays, disassembly operation simulation of industrial products finds a strong interest in interactive simulations through immersive and real-time schemes.
However, the available disassembly evaluation methods today seldom make disassembly as the preferred end-of-life solution for the reuse of parts or components in an economically sustainable way for lower value products. In virtual environment, for instance, a human model is often involved in a digital mock-up (DMU) model for assembly/disassembly evaluation. However, it has limited application areas because of its high cost investment.

In this context, to meet some problems, a part of which were evoked here above, we define the objective of our research as follows: "*The research aims to define novel and efficient methods, models and tools allowing designers and industrials to take into account the constraints of disassembly operations during the initial stage of product design and/or to automatically generate the selective disassembly sequences and their evaluation as well*".
Disassembly operations covering a broad range of the PLC[*], our research is particularly attached to answer the following questions:

- How to define and formalize the disassembly of a product?
- What are the product characteristics which affect its desassembility?
- How to obtain the minimum number of possible disassembly sequences in the case of selective disassembling?
- How to evaluate them in ergonomical and technological point view?
- Which kind of criteria should we propose for this purpose?
- How Virtual Reality may help in this way?

Research contribution

In this context our research attempts to develop new comprehensive methodology and tools enabling to establish a simplified model for the generation of disassembly sequences and their evaluation in a VR environment. Our objectives are not only to reduce the complexity of disassembly sequences generation model, but also to evaluate the disassembly sequences in virtual

[*] Product Life Cycle

reality environment (VRE) via automatic ergonomic evaluation.

In the first place, the aim of this thesis is to develop a new method for generating selective disassembly sequences. When disassembling, it is important to eliminate the components which are unrelated to the target components prior to sequence generation. In order to address this configuration, this thesis presents a method for generating the feasible disassembly sequences for selective disassembly. The method is based on the lowest levels of a disassembly product graph. Instead of considering the geometric constraints for each pair of components, the proposed method considers the geometric contact and collision relationships among the components in order to generate the so called Disassembly Geometry Contacting Graph (DGCG). This graph is used for disassembly sequence generation thus allowing the number of possible disassembly sequences to be reduced by ignoring any components which are unrelated to the target. The method is applied for automatic generation of selective disassembly sequences for mechanisms with different degrees of complexity. The disassembly simulations can be performed either from an automated or interactive point of view using standard computer equipment or through immersive and real-time simulation schemes. In order to address this diversity of configurations, a simulation framework was developed integrated in a Virtual reality environment thus allowing generating the minimum number of possible disassembly sequences.

As previously said, the available disassembly evaluation methods today seldom make disassembly as the preferred end-of-life solution for the reuse of parts or components in an economically sustainable way. In recent years Virtual Reality interface has been wildly used to simulate various processes and in particular assembly/disassembly operations during the initial stage of product design. Thus, in the second time, a method for disassembly operation evaluation by 3D geometric removability analysis in a Virtual environment is proposed. It is based on seven new criteria which are: *visibility of a part*, *disassembly angles*, *number of tools' changes*, *path orientation changing*, *sub-assembly stability*, *neck score* and *bending score*. All the criteria are presented by dimensionless coefficients automatically calculated thus allowing evaluating disassembly sequences complexity. For this purpose, a mixed virtual reality disassembly environment (VRDE) is developed based on Python programming language, utilizing VTK (Visualization Toolkit) and ODE (Open Dynamics Engine) libraries. The framework is based on *STEP*, *WRL* and *STL* exchange formats. The analysis results and findings demonstrated the feasibility of the proposed approach thus providing significant assistance for the evaluation of disassembly sequences during Product Development Process (PDP).

Thus, this manuscript concerns:

- The development of a new method for generating selective disassembly sequences capable of minimizing their number, as previously mentioned. Building the DGCG is based on a procedure consisting in: *Contact identification, Set of directions of removal* (SDR) and *Collision detection*. For this purpose three cases called *micro units*, which consider all the possible situations of relationships among the components in the *DGCG*, are proposed namely: Transition from No SDR (*NS*) to Collision (*C*); Transition from Collision (*C*) to Collision (*C*) and Transition from no SDR (*NS*) to no SDR (*NS*)

• The development of a demonstrator, to generate the possible disassembly sequences in the case of selective disassembly and their evaluation as well.

• Integration into the product development process (PDP) by: identifying contacts, Set of directions of removal and collisions detection; using different types of subsequent simulations based on the requirements imposed by a PDP.

Application area(s) of this research

Two major aspects are addressed here:

- Concerning the generation of selective disassembly sequences, the results of this study may be useful for designers and industrials, allowing them to take into account of the constraints of disassembly operations during the initial stage of product design and/or to automatically generate the selective disassembly sequences, which cover a broad range of PLC.

- Concerning the efficiency evaluation of disassembly sequences, the thesis provides a new way to assess the difficulty of disassembly sequences in VR environment instead. The resulting score values of the proposed criteria are a decision taking aid for designers to assess disassembly sequences efficiency evaluation for a product.

The scientific repercussions of this work relates in particular to disassembly operation modeling and its integration with the PDP in the sector of manufacturing industry. From an industrial point of view, it is a question to bring brief replies to the current industrial needs (Renault, EADS,..) concerning the modeling and the simulation for disassembly operations and their evaluation.

Structure of the memory

The manuscript, retracing our three years of research activities, consists of five chapters.

In Chapter 1, the state of art concerning the context of disassembly sequencing is presented. Some relevant methods, models and searching algorithm are discussed and analyzed. A critical assessment concludes this chapter evoking the need for new models and corresponding developments.

The context and previous work of Assembly/Disassembly operation simulations in virtual reality systems are introduced in Chapter 2. Relevant usual systems, devices, analysis methods and challenges related with VR technique are presented, followed by a critical analysis and assessment. In Chapter 3, a new method for selective disassembly sequences generation, based on the least levels of disassembly product graph, is proposed. Instead of considering the geometric constrains for each pair of components, the proposed method considers the geometric contact and collision relationships among the components in order to generate the disassembly geometry contacting graph (*DGCG*) for disassembly sequences generating.

The key technologies and devices of the created virtual environment for disassembly sequences generation are detailed in Chapter 4. A virtual reality disassembly environment (VRDE) is presented based on Python programming language, utilizing mixed VTK (Visualization Toolkit) and ODE (Open Dynamics Engine) libraries.

In chapter 5, a new method for disassembly evaluation by 3D geometric removability analysis in VR environment is proposed. It introduces some new parameters such as: *visibility of a part*, *disassembly angles*, *number of tools' changes*, *path orientation changing*, *sub-assembly stability*, *neck score* and *bending score*, thus allowing performing and evaluating disassembling task in a VR environment.

This PHD thesis work was realised in *Information system design Robust Products* (ISDRP) team of G-SCOP Laboratory under the co-direction of Dr. Peter MITROUCHEV associate professor at University of Grenoble, France and prof. Lixin LU from the Department of Mechanical Manufacturing & Automation, Shanghai University, China. The experimental part, tests and simulations were conducted in Gi-Nova Plateforme Technoligique, Systèmes de Production, AIP-PRIMECA, Dauphiné-Savoie, Grenoble, France.

Acknowledgment: *The research leading to these results has received funding from the European Community's Research Infrastructure Action - grant agreement VISIONAIR 262044 - under FP7/2007.-2013 (http://www.infra-visionair.eu/)*

Chapter 1

Disassembly sequences generation methods

This chapter analyzes the results and remaining problems of existing research in the field of disassembly sequencing modeling. Detailed classification is done in the beginning of this chapter. Then, different methods such as: interactive and automatic are introduced. Automatic methods seems being the ideal for the sequences generation, some related algorithms based on Artificial Intelligence (AI) methods are also presented. These analyses highlight the need of appropriate method for reducing the complexity for product sequences calculation.

1.1 Introduction

Disassembly processes are studied for a number of reasons as they cover a broad range of the Product Life Cycle (PLC) regarding operations of disassembly during: production process, product maintenance and finally at the end of product life cycle. Disassembly may be defined as: *a systematic method for separating a product into its constituent parts, components and subassemblies* [Gun 01].

Disassembly sequencing involves searching all the possibilities to disassemble a product and often the selection of the optimal solution out of these. For the company, the improvement of the recyclability performance of the products is becoming an integral part of their product development process (PDP). Let us note that there are two revolutionary key concepts related with the disassembly applications. The first one is the responsibility of the manufactures for the whole life of a product integrating assembly and disassembly in the same time. The other one is that disassembly is based on the concept of "*selling use*" instead of selling products.

As mentioned in the general introduction, completed disassembly is not the preferred method, therefore, this thesis focus on the selective disassembly instead. Thus, in the next of this chapter, the detail survey of the presently available literature on the disassembly is presented involving existing methods, models, algorithms and tools.

1.2 Generality about disassembly

In the very beginning, the problem for generating disassembly sequences has been addressed by engineers, while aiming at the investigation of assembly process. In that time, the disassembly is assumed as the reverse of assembly. In fact, disassembly being the process of separating components can be classified according to the purposes of disassembly and the way for performing of such disassembly. Consequently, it is not completely the reverse of the assembly.

1.2.1 The purposes of disassembly

According to the life-cycle scenario of the product, the needs for disassembly include different stages such as: maintenance, repairing, remanufacturing, recycling and disposal. As mentioned in the General introduction, within the disassembly sequences, it is possible to distinguish between complete disassembly and selective disassembly. Complete disassembly involves disassembling all the components of a complex object. It has been mainly studied as a solution to assembly planning, since the reverse of a disassembly sequence is, in fact, an assembly sequence according to Gottipolu et al. [Got 03]. A complete disassembly is rarely the optimal solution owing to the high costs of the disassembly [San 02].

Selective disassembly, which requires only a portion of an assembly with high value to be disassembled, suggests that the most economical assembly sequences is not the most economical disassembly sequences [Sri 99a]. Therefore, the differences between assembly and disassembly analysis make a separate study of product disassembly important. Selective disassembly is usually more appropriate for demanufacturing application, such as maintenance, repairing or recycling.

1.2.2 Methods for disassembly

Prior to present the new method for selective disassembly (Chapter 3) some different types of disassembly methods are presented here below.

From a disassembly view point, there are two main types of disassembly methods.

- "destructive disassembly", in which a component is removed from the product previously disassembled, by destroying or damaging some other components of the product.
- "no-destructive disassembly", in which each one of the components can be removed without affection any of the others [Pom 04].

From the purpose of the disassembly and according to the end life-cycle scenario of the product, the no-destructive disassembly is more useful in some processes such as: maintenance, repairing, remanufacturing, recycling and disposal.

The "no-destructive disassembly" method can be sub-classified also such as:

- direct and indirect disassembly,
- sequential and parallel disassembly,
- monotonic and non-monotonic disassembly.

Direct and indirect disassemblies are based essentially on the number of components that have to be removed in order to reach the target component. If the target components can be removed without removing other components, it is the direct disassembly. Otherwise it is indirect one.

Sequential disassembly is based on the number of components that are disassembled at a time [Kan 01]. For example, if each time only one component can be disassembled it is called as the sequential disassembly. Otherwise it is parallel disassembly, which means that in the sub-assembly, the components can be disassembled in a single group instead.

Monotonic disassembly depends upon whether it requires moving out the total of the component from the assembly. For example, if in a room, we can take out a bed through opening the door or disassembling the door. If it needs to disassemble the door, we call it as monotonic disassembly. Instead, if it is just necessary to open the door, but the components are not disassembled completely from the assembly, we call it as non-monotonic disassembly.

As previously said (Section 1.2.1), that there are two kinds of disassembly planning which are:

- complete disassembly
- selective disassembly.

Complete disassembly involves disassembling of all the components of an assembly, but it is rarely the optimal solution owing to the high costs of the disassembly. Alternatively, selective disassembly is usually more appropriate for manufacturing applications, such as maintenance, repairing or recycling.

Concerning the depth of disassembling, there are:

- total disassembly,
- partial disassembly (selective / targeted).

Concerning the nature of disassembly:

- linear disassembly,
- parallel disassembly.

Other classification defines the methods for generation of disassembly sequences in:

Exact methods, which are:

- Analysis of modularity [Kuo 00a, Kuo 01],
- Branch and bound [Gun 01, Zha 10],
- Wave propagation [Sri 98, Sri 00, Mas 03, San 02],

Approximate methods, which are:

- heuristics, [Kuo 00b]
- metaheuristics :
    - Genetic algorithm [Sri 98, Kon 06a, Giu 07, Tse 09],
    - Ant colonies [Wan 03, Tri 09],

Methods based on the feedback:

- Case-based reasoning [Zei 97],
- Reasoning Knowledge Base [Vee 02],
- Learning technique [Ale 11].

1.2.3 The disassembly character involved in the thesis

In this thesis, considering disassembly as an aim in itself we would like to highlight its distinction from assembly. There are essential differences that have to be pointed out here. First, selective disassembly is often preferred as the research target in the disassembly field as stated by Lambert in [Lam 03], because not all the assembly process can be reversible. Therefore, the selective disassembly will be our research target. Second, the destructive disassembly is not our concern, even though; it may be useful for some valuable component recycling. Third, monotonic disassembly is not related within our work.

1.3 Representative disassembly methods

Let note, that disassembly sequencing is listings of subsequent disassemble actions, which involves the search for all possible disassembly sequences. There exists extensive research on disassembly sequences analysis and the disassembly sequence optimization. For the sequences generation, researchers have suggested several approaches to determine the disassembly/assembly sequences. The methods for the disassembly sequences generation can be classified into three groups as *interactive*, *automatic* and *Artificial Intelligence* (AI) methods.

Interactive approaches require extensive user input usually in the form of answering questions, whereas automated approaches can be mainly used to generate disassembly processes for products with simple component configuration and geometry. In this thesis automated and interactive techniques are combined, using virtual reality environment (VRE) to generate and evaluate selective disassembly sequences in the process of product design.

Disassembly sequences being a key element of the simulation of a product disassembly process, this chapter aims to present the main approaches adopted for disassembly sequences generation. Thus, a review of some deterministic and stochastic approaches is presented in the next. Note that deterministic methods produce sequences, while stochastic ones analyze the sequence generating process. Stochastic methods have to cope with the combinatorial complexity in most disassembly process simulations by using only geometric data as a starting point or that do not bind the technological parameters of a product with its digital mode [Iac 10].

1.3.1 Interactive methods

Interactive approaches can be used to handle with complex assemblies because information is not gathered from the geometry of the components. Instead, the information is gathered from designers. These approaches need that the designer or the operator should be very familiar with the product (assembly). Note that time and knowledge involved for answering questions about the assembly are the main disadvantage of the interactive approach.

Bourjault was amongst the first, who in 1984 [Bou 84] proposed the definition of the so called *assembly precedence relations* (APRs). Based on these relationships, a *liaisons diagram* was proposed by him to represent the assembly. In this liaisons diagram, series of questions were needed to be answered by the users with '*yes or no*'. Later, Homem and Sanderson [Hom 91] applied the '*cut-set*' analysis method in the assembly sequences planning and designed three simple rules. He count that the queries could be reduced by 95%, however, *111* questions in the sequences planning need to be answered for only *11* part assembly. In order to reduce the number of the questions at the same year, Baldwin et *al*. [Bal 91] developed a method using the '*what*' questions instead of '*yes or no*'. However, obviously it is more difficult for user to answer the '*what*' questions correctly, especially when the products are relatively complex and the users are not familiar with the products. Other authors Johnson and Wang [Joh 98] enhanced the '*cut-set*' analysis and a man-computer interactive method supported by assembly CAD draft, where the user has to indicate the interference parts if collision occurred in the disassembly.

The interactive approaches base their reasoning process on the *liaisons* diagram. However, because the liaisons diagrams are too simple to get enough information for the sequences analysis, it requires user to answer a number of questions to tell the system how the assembly looks like. Let us note that, this method, classified as indirect approach by [Su 07], is rarely used nowadays.

A deterministic approach for disassembly sequences generation based on the observations of industrial planning assemblies is proposed in [Bar 04]. It uses simultaneously Design for Assembly (DFA) and a conventional design process approach. The proposed tools and techniques allow generating disassembly sequences and defining assembly configurations, including assistance to build the sequence, choose the most relevant documents and to define connection parameters. A constraints approach (CSP) is used to confirm that the resulting assembly sequence would be feasible thus providing a quality assessment of the sequence.

1.3.2 Automatic methods

Today automatic methods are considered as ideal way for sequences generation. They can use the relationships among the components for the disassembly path calculation. However, for the disassembly sequences generation, the model has to be with relatively simple component geometry. There are lots of researches working on this field in order to reduce the computer resource for the generation of the disassembly sequences. Some basic concepts or algorithms should be clear in this domain. For this purpose some existing relevant and important works, in our knowledge, are presented here below.

(a) Graph for Disassembly Model

With regards to automatic methods there are two major steps for disassembly planning generation, which are disassembly model creating and disassembly sequences generating. The model is usually presented by graphs and the sequences generations are based on the matrices which are converted from the graphs. Graph theory in the field of mathematics is very common way to show pair-wise relations among features in the graph. In order to model product's topology and geometry information, a number of graph based modeling strategies are used for the disassembly sequences generation, such as *And/Or* graph, *Liaison* graphs, etc.

- And /Or graph [Hom 90, Hom 91, Lee 94, Kan 01, Got 03, Zhu 13]

In order to present all possible disassembly sequences, many methods are proposed. Among them, the And /Or Graph has been widely used to represent disassembly sequences (Fig. 1.1).

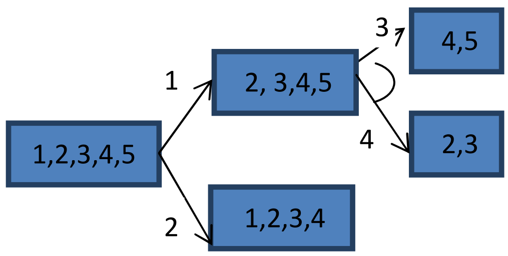


Figure 1.1 An example of AND /OR graph

In And / Or Graph, each node can be a product, a component or a subassembly the edges being the links among them. For disassembling a part if other components are involved, the link edges form the And relations. If more than one paths can reach the target part, this will form the Or relationship. For an And/Or graph, *G=(N, D)*, (where *N* stands for the nodes that denote a product, sub assembly, or part and *D* stands for hyper arcs, means the sets of feasible disassembly operations) if each node can have *m* $(m \geq 1)$ possibilities for disassembly, it forms the 'Or' relationship in the graph. If one operation disassembles into more than one node in the precedent graph, it forms the And-relation. As shown in Fig.1.1, *N={(1,2,3,4,5), (2,3,4,5), (1,2,3,4), (4,5*), (2,3)} and *D={1,2,3,4}*. For *D1*, *N=(2,3,4,5)*, part *1* has to be disassembled. For D2, *N=(1,2,3,4)*, part *5* has to be disassembled. Therefore this is the "Or" relationship. For *D3*, after getting the parts *4* and *5*, the parts *2* and *3* will be moved automatically. Therefore this is the "And" relationship. One of the advantages of the And/Or graph is that it requires relatively small space for the storage. However, there are some information missing in the And/Or graph. For example, the relationship between operations *D2* and *D3* is not clear in Fig.1.1. Do we need to perform operation D2 before the D3? Note, that this approach does not contain any information about the subassemblies.

- Precedence Graph [Joh 98, Gun 01, Lam 08]

  Precedence Graph methods, being a part of automatic methods, are focusing on the precedence relationships, which aim at the automatic generation of disassembly sequences. It is derived from the task precedence graph, which is commonly used in the task planning issues. Different with the *And/Or* graph, the operations are presented by nodes. The arcs are directed arcs pointing from one operation to another. The operation where the arc points from should be performed before the operation where the arc point to.

  As shown in Fig.1.2, arcs indicate the precedence relationships that exist between two subsequent operations. In this example, it is easy to distinguish the *18* precedence relationships.

  In [Joh 98] authors used precedence graph to represent the products' structure, which is hierarchically organized according to a Bill of Material (BOM). Some details for the various operations are considered in the costs and the profit of the disassembly operations. Compared to *And/Or* graphs, the disassembly precedence graph has less nodes. For this model, its *cpu* time is still unmanageable for solving problems, as stated in [Lam 08].

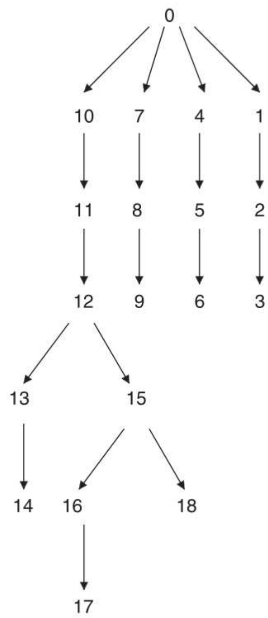


Figure 1.2 Disassembly precedence graph [Joh 98].

- Extended process graph

  Proposed by Kan *et al.* [Kan 01] extended process graph is obtained by transforming the And/Or graph using precedence relations. In the graph, a path from source to sink represents a disassembly sequences as well as the disassembly level (Fig. 1.3). Each node represents a source, a sink, an operation, a choice or a separation. In Fig. 1.3, a filled circle ● means that there are more than one possible ways to disassembly a subassembly. An empty circle ◯ means a separation node, each of which contains more than one part and requires more disassembly operations. All of them can be transformed from *Or* and A*nd* relationships in the And/Or graph respectively. Each solid arc represents a precedence relation between two operations. There are two dotted arcs, including the unidirectional arcs, which link each operation node to the sink node and the bidirectional dotted which represent two possible orders between two operations. In other words, the subassembly can be disassembled in parallel and there is no precedence relation between the operations.

  As seen from Fig.1.3, if the path can satisfy the precedence constrained by the solid arcs, the path is feasible for selection.

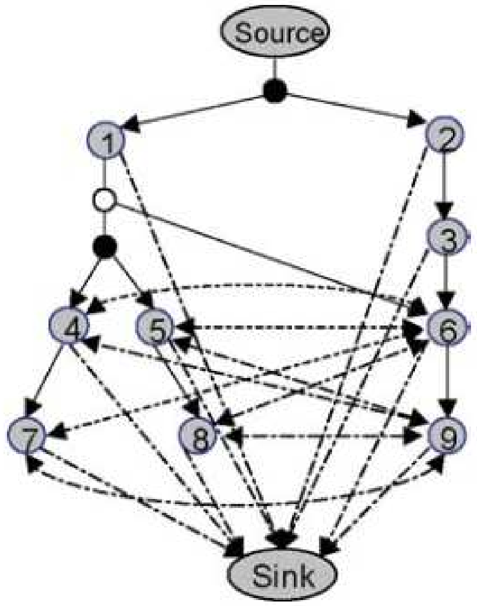


Figure 1.3 Extended process graph [Kan 01]

Note, that this graph successfully extends the method from the precedence graph to And / Or graphs (because of the increased number of nodes in the And/Or graph). However, the problem can be solved only at the expense of unmanageable *cpu* time even if moderately complex problems are considered as was discussed by Lambert in [Lam 08].

- Liaison or connection graphs

  First developed by De Fazio and Whitney [Faz 87], liaison or connection graphs, are great different from the precedence graphs. Let us remember that, in a precedence graphs, the relations are not the components which are physically connected. The relations are the precedence order between the components. In a liaison graph, circles (vertexes) represent components in the assembly and edges physical connections between them. Therefore, the liaison analysis is performed by examining the geometric connection between the parts in the product, and the liaison diagram consists of a network of nodes and lines that represent the parts and the relationships between parts. If n is the number of parts, the possible number of liaisons ranges between $(n^2-n)/2$ and $(n-1)$.

  For each liaison, it is necessary to have the precedence rules in order to determine the optimal level of disassembly either for a single or multi objective criterion. For example Kara et *al.* [Kar 06] used liaison and precedence rules to generate the selective disassembly sequences of a washing machine as shown in

Fig1.4. Lee et *al.* [Lee 94] used the abstract liaison graph for the purpose of sub-assembly identification to aid in automatic generation of assembly sequences. Later Dong et *al.* [Don 06] tried to use the liaison program in order to create a hierarchical attributed liaison graph for disassembly sequencing. Recently, in, [Ric 13] authors transformed a liaison graph in to weighted liaison graph (WLG) to show the disassembly precedent relations amongst all the components.

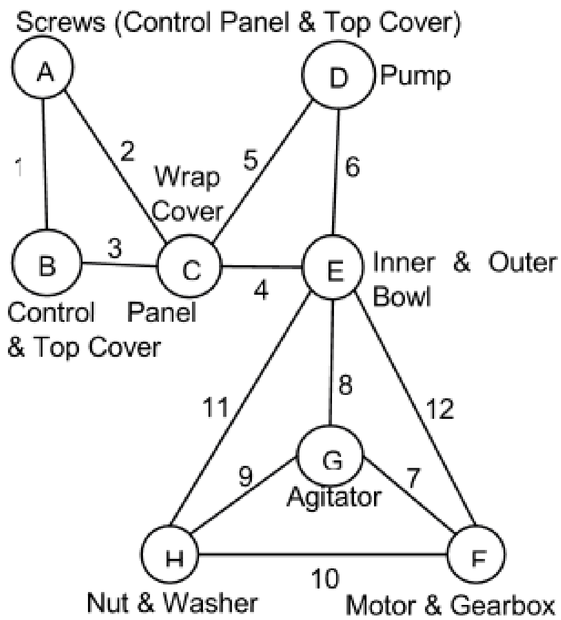


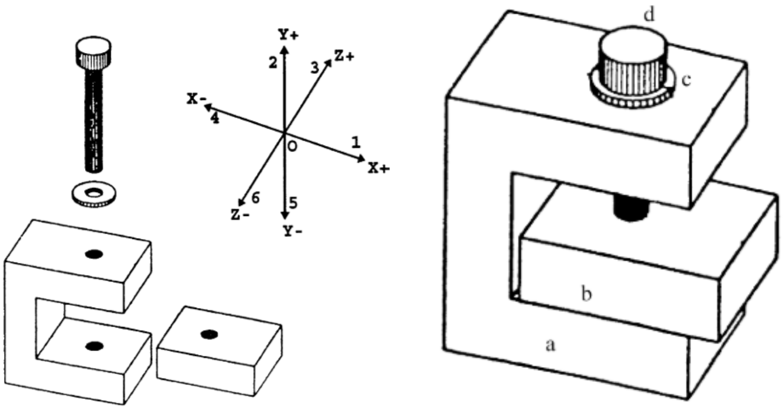
Figure 1.4 Extended process graph [Kar 06].

However, the liaison graph cannot work itself for the sequences generation. The liaison sequence graph gives liaison sequences instead of assembly sequences. The *states* do not present a set of parts, but a set of relations between them. The state has to work with precedence rules for each liaison which means that the questions for each liaison have to be answered for all the liaisons prior to sequences generation. The precedence rules are used as input formation for the disassembly sequences generation. However, there are two major drawbacks related with this methodology: the liaison graph has to be established by very experienced designer and the product must be relatively simple.

(b) Matrix analysis

For disassembly based on *Matrix analyses* method, there are two major steps which are: creating disassembly model and generating disassembly sequences. The model is usually presented by graphs. The sequences generations are always based on the matrices which are converted from the graphs [Wil 94]. Combined use of graphs and matrices has been proposed by many authors [Kuo 00b, Got 03, Smi 11, Smi 12, Ou 13]. The disassembly graph can be represented by a transition matrix, in which the columns correspond to the possible disassembly actions, the rows to all the possible subassemblies or components. Note, that for automatic calculation, the matrix analyses method is one easy way to generate the possible disassembly sequences. In order to better understand the converting process from the graph to matrix, the relevant works of two researchers are presented with some details here below.

i) Assembly sequences table (AST) [Got 03]



(a) Exploded view (b) Assembled view

Figure 1.5 Example of assembly represented in tree-dimensional space [Got 03]

Definition:

One assembly in relational model is a two-tuples <P, U>, where $P = \{P_1, P_2 \ldots, P_n\}$ is the set of symbols corresponding to one part (no two symbols' correspond to the same part) and $U = \{U_1, U_2 \ldots, U_m\}$ the relations between components, m being the number of component ordered pairs with $U_i = <P_a, P_b, C_{ab}, T_{ab}>$. The contact function C_{ab}, represents the contacts between the components a and b. The translation function T_{ab}, represents translational motion between components a and b.

Function $C_{ab=}$ $(C_1, C_2, C_3, C_4, C_5, C_6)$, presents the six directions of contacting information, where $C_i = \{0, 1\}$. Concerning the directions, they can be the coordinates ($x+$, $y+$, $z+$, $x-$, $y-$, $z-$) as shown in Fig.1.5 (a).

If C_i=0, there are no contact in the direction i. If C_i=1, there are contact in this direction. In the same way, the translation function $T_{ab} = (T_1, T_2, T_3, T_4, T_5, T_6)$ represents the translational motion between component a and b in six direction, where $T_i = \{0, 1\}$. If T_i=1, indicates the part b has freedom of translational motion with respect to the part a in the direction i.

Thus, according to this rule, it is very easy to build the functions C and *T* between every two components in the product. For the assembly in Fig. 1.5(b), the *C* and *T* functions are built as shown in Table 1.1.

Table 1.1 Function C and T for the assembly shown in Fig.1.5 (b) [Got 03]

Pair	*C*-function						*T*-function					
	C_1	C_2	C_3	C_4	C_5	C_6	T_1	T_2	T_3	T_4	T_5	T_6
(***a***, ***b***)	0	0	0	1	1	0	1	0	1	0	0	1
(***a***, ***c***)	0	0	0	0	1	0	1	1	1	1	0	1
(***a***, ***d***)	1	0	1	1	1	1	0	1	0	0	0	0
(***b***, ***c***)	0	0	0	0	0	0	1	1	1	1	0	1
(***b***, ***d***)	1	0	1	1	0	1	0	1	0	0	0	0
(***c***, ***d***)	1	0	1	1	1	1	0	1	0	0	0	0
(***b***, ***a***)	1	1	0	0	0	0	0	0	1	1	0	1
(***c***, ***a***)	0	1	0	0	0	0	1	0	1	1	1	1
(***d***, ***a***)	1	1	1	1	0	1	0	0	0	0	1	0
(***c***, ***b***)	0	0	0	0	0	0	1	0	1	1	1	1
(***d***, ***b***)	1	0	1	1	0	1	0	0	0	0	1	0
(***d***, ***c***)	1	1	1	1	0	1	0	0	0	0	1	0

From our understanding, *C* function determines the local feasibility for removing a component, while the T function determines the global feasibility.

In this work, the six directions (x+-; y+-; z+-) are taken into consideration. Functions *T* and *C* are both *1x6* binary functions. PADL_2 software also was developed by the authors in order to extract these functions. Note, that the presence or absence of the contact cannot guarantee the free collision motion of the components. So, if the subassembly (*a, b*) is supposed to be a set, it needs to be added to another component (*c*). Thus, sets (*a, c*) and (*b, c*) should be both taken into consideration. If a contact exists in any direction and component *c* can be in contact with *a* or *b*, consequently *c* could be added in the subassembly. So the "or" operator is used in this situation. If *c* is planned to be disassembled, it has to be free related with both *a* and *b*, so the "and" direction is used in this situation. Finally, each assembly tasks is connected with the weight factors for the evaluation of the assembly plan.

Note, that Assembly sequences table (AST) method is a simple way to evaluate all the possible assembly sequences and provides a useful method to simplify their evaluation which has the principles for matrix calculation considering the six directions of assembly and disassembly. However, its limits are that *C* and *T* functions only take 6 directions into consideration. Since the logic works with Cartesian coordinates, it would be better to take all the *C* and *T* functions in cylindrical and spherical coordinate systems. As previously said, this method could be also used for disassembly evaluation. We think that it just needs to check *T* function and the stability of the subassemblies. In our opinion, a new method is needed to be proposed in order to evaluate the stability of the subassemblies (see details in § 5.2.2, Chapter 5) prior to disassembly sequencing, which is an aim of our thesis.

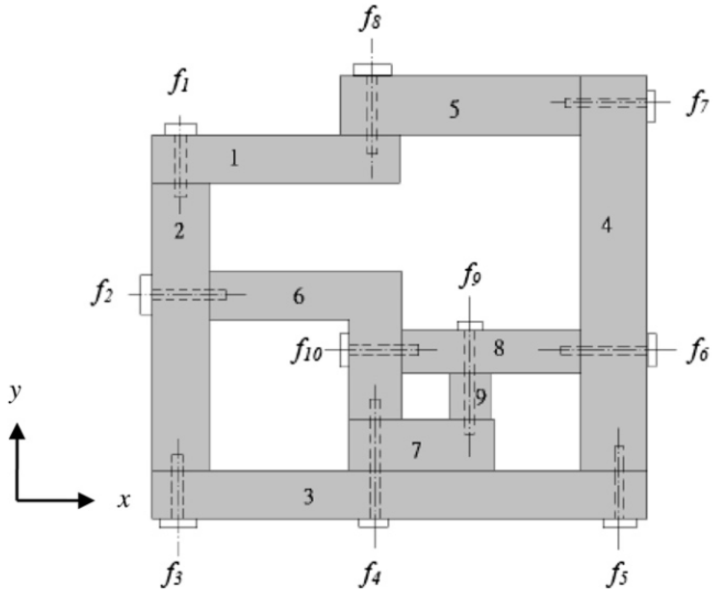
ii) Disassembly sequence structure graph [Smi 11, Smi 12]

A new disassembly sequence structure graph (DSSG) method was proposed by Smith and Chen in [Smi 11] to improve the solution quality, minimize the model complexity and reduce search time. Note, that graphs containing more information improve solution quality, while those containing less information reduce searching time. The DSSG model contains one graph, which only includes the nodes and links needed to remove the target components, and five matrices. Two of them are contact constraint matrices, two others are motion constraint matrices and one is a projection matrix. The authors divide the parts in a product into two categories as: components and fasteners. They assume that a fastener must be removed from a component, but a component cannot be removed from the fasteners. Thus, contact constraint matrix includes the contact constraint for components and contact constraint for fasteners. Motion constraint matrix includes the motion constraint for components and for fasteners. The projection matrix determines the removing direction of the target component with the rule of the least numbers of parts to remove.

The contact constrain (*CC*) matrix for the components of an assembly is:

$$
\begin{array}{c}
\begin{array}{cccc} +X & -X & +Y & -Y \end{array} \\
CC = \begin{bmatrix} CC_1 \\ CC_2 \\ CC_3 \\ CC_4 \\ CC_5 \\ CC_6 \\ CC_7 \\ CC_8 \\ CC_9 \end{bmatrix} = \begin{bmatrix} 0 & 0 & f_1, f_8, 5 & 2 \\ 6 & f_2 & f_1, 1 & f_3, 3 \\ 0 & 0 & 2,4,7 & f_{3,} f_4, f_5 \\ f_6, f_7 & 5,8 & 0 & f_5, 3 \\ f_7, 4 & 0 & f_8 & 1 \\ 8 & f_2, f_{10}, 2 & 0 & f_4, 7 \\ 0 & 0 & f_{9,} 6,9 & f_4, 3 \\ f_6, 4 & f_{10}, 6 & f_9 & 9 \\ 0 & 0 & f_9, 8 & 7 \end{bmatrix}
\end{array}
$$

For a product with *N* components, the *CC* matrix has *N* rows. For this matrix, columns represent the *+X, -X, +Y, -Y* direction. For example, CC_1 represents the component *1* in the Fig. 1.6.



legend: *f*- fastener

Figure 1.6 Example of assembly [Smi 12]

In the *+X*, *-X* directions, there are not contact components. In the *+Y*, *-Y* directions, the contact components are f_1, f_8 , *5* and *2* respectively. The motion constrain matrix is established in the same way.

The projection matrix records the blocking components of each component which are used for the optimization of the disassembly directions. The fewer the blocking components, the better the direction is. However, the Matrix analyses method is limited because it only takes into account a limited number of Disassembly/Assembly directions/translations (four or six) without rotations. Another difficulty is to build the matrixes automatically for complex assemblies. For building all the matrixes related with the product and finding the relationships among the components, a great search time is necessary because the number of possible disassembly sequences increases exponentially with the number of parts in the product.

1.4 Searching algorithms

In order to generate the feasible disassembly sequences, after establishing the representative model (graph, matrix or others), a systematic analysis, based the time saving algorithms, is necessary to be performed. Some of the most useful algorithms are presented in the next.

1.4.1 Heuristic searching

Let us remember, that in computer science, a heuristic is a technique designed to solve the problem more quickly when conventional methods are too slow, or to find an approximate solution when conventional methods fail to find an exact solution. Heuristic are based on a predefined set of rules used in the sequences generation of Disassembly or Assembly. The problem of finding optimal or even near-optimal disassembly sequences is known to have an exponential computational complexity.

1.4.2 Wave propagation approach

As we are aware, the most popular and applied determinist method for selective disassembly is the wave propagation algorithm [Sri 99a, Sri 99b, Sri 00]. This selective disassembly is particular convenient for maintenance and disassembly for recycling. Selective disassembly analysis of a product with *n* components and *s* selected target components ($s<n$), determines a disassembly sequence for *s* components considering the minimum number of components to remove. The wave propagation algorithm automatically reduces the computation complexity *O (n2)* by finding the shortest way to take out the target component from an assembly.

There are three assumptions for the wave propagation approach.

- First, the relative motions of components are determined without considering the tools and the fixtures.
- Second, the assemblies are assumed to be polyhedral.
- Third, the components have single linear motion allowing them to be removed from the assembly.

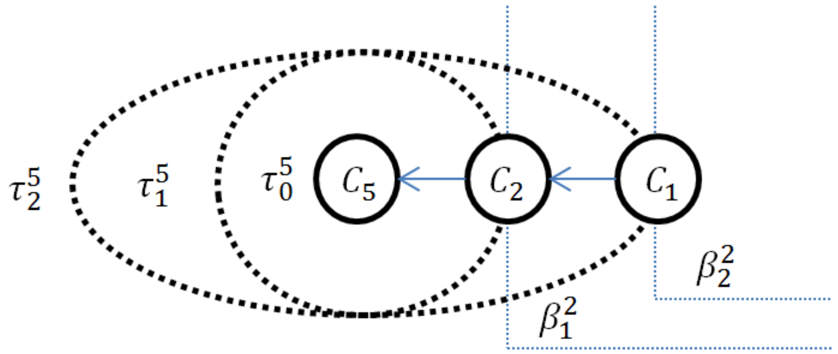


Figure 1.7 Illustration of the wave

Two types of disassembly waves are defined. One is τ wave source from the targets which determines the disassembly ordering. The other one is β wave from the boundary of the assembly which determines the minimum number of components to be disassembled in order to reach the target. Based on these disassembly waves a disassembly sequences is automatically generated by the intersection events between them. For every component *c*, the waves τ^c and β^c are defined. For every time step (from *t=a* to *t=a+1*), τ^c propagates by one wave front. As shown in Fig. 1.7, at *t=1*, β^2 wave propagates from $\beta_0{}^2$ to $\beta_2{}^2$, where *C2* is disassembled after removing *C1*. Wave disassembly uses modeling component with reduced mobility only to translations and intersections of translational directions attached to each side of the contact surfaces to locally validate or not a disassembly operation. The components are geometrically defined by the faceted models and the contact areas between the components are considered as input data.

Wave propagation method allows reducing the analysis of expensive-typically exponentially. However, it has two major drawbacks. One is that it does not show how to build the graph based on the real relationships among the components in an assembly. The second drawback is its limitation to generate efficient and optimal sequences for disassembly of complex products as mentioned in [Smi 12].

1.4.3 Dijklstra's algorithm with heap-based priority queues [Gar 04]

This algorithm, proposed by Garcia et *al.* [Gar 04], is based on the precedence graph where each node represents a single component of the assembly. Its computational complexity is of *O(nlog n),* when $s<<n$, and *O(sn)* time when $s\cong n$. A simple example of an assembly containing 9 components, where 3 of them are exterior (C_1, C_8, C_9) is shown in Fig.1.8.

The directed edge from a certain node C_1 to another node C_2 indicates that if C_2 has already been disassembled, the C_1 can also be removed from the assembly. Component C_5 has *OR* relationship with C_4 and *AND* relationship with both C_6 and C_7. Which means that C_5 can be removed right after extraction either C_4 or both C_6 and C_7. The graph can be simplified if C_1 is not exterior. In this case all the edges downwards in Fig. 1.8 would disappear.

The algorithm consists in four stages:

- Computing the shortest path between each exterior node and the rest of nodes of the precedence graph.
- For each selected component, determining the shortest paths to its closest exterior components, by taking into account the edge labels computed in the first stage.

- Merging the partial disassembly sequences obtained above.
- Sorting the sequences in descending order of selected components.

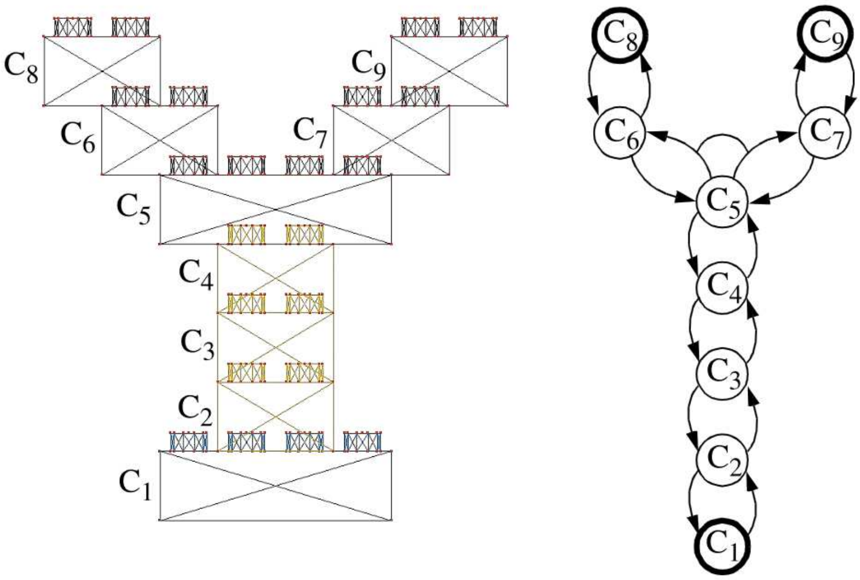


Figure 1.8 Assembly and its corresponding precedence graph [Gar 04]

The algorithm computes the minimum distances from the exterior components of the assembly to the rest of the components. Then, for each target, a set of partial disassembly sequences is obtained by finding minimum spanning trees in the precedence graph. The solution is found by sorting sequences from the head until all the selected components have been considered. However, the algorithm has a disadvantage. It concerns the determination of the external components. Note, that for an assembly, it is difficult to determine which component should be the external as there are many possibilities for that. (components C_8,C_9,C_1 in Fig. 1.8,) Note, that the author did not mention the method for building the precedent graph either. As from our knowledge, there is no automatic way to do it.

1.4.4 Artificial Intelligence (AI) methods

AI methods focus on detecting the best sequences when a combinatorial explosion of sequences takes place. In order to obtain one or several sequences based on the profit of each sequence, the profit of all feasible sequences has to be calculated, which leads to unacceptable computing time. AI methods try to replace the traditional method with an objective to reduce searching time by searching the best sequences without analyzing all the possible alternatives. In the past years, AI techniques such as: Petri Nets, Genetic Algorithms, Neural networks, Fuzzy set, Bayesian networks have been used especially related with disassembly sequence optimization.

- Petri nets (PNs) is the widely diffused method which has the advantage of taking into account of many factors (time, economic value, environment aspects etc.) for the disassembly planning and scheduling. Four primitive elements (*tokens*, *places*, *transitions* and *arcs*) and certain rules are involved for controlling the operations. The tokes are conceptual entities appearing as small solid dots. The places are shown as circles and stand for the locations where objects await the processing or the condition of objects. The transitions are shown as bars or rectangles which present processes, events, or activities that may occur. The arcs present the paths of objects in the system.

  In our knowledge, the first PNs used in disassembly, was presented by Zussman et *al.* [Zus 95], who developed a disassembly Petri net (DPN) through the notion of an inverted assembly PN. Later, Moore et *al.* [Moo 98, Moo 01], presented an algorithm using a PN-Based approach to automatically generate the disassembly process plan (DNP), which considered the simple AND, OR, Complex AND/OR, and XOR relationships as shown in Fig. 1.9. Shiung Hsieh in [Hsi 08] set up a greedy algorithm to find a nominal optimal solution for the aforementioned problem and study the fault tolerant properties of the nominal system. Recently, Kuo C. Tsai [Kuo 13], proposed a PN approach to consider the economic value and environment pacts on the disassembly and recycling processes.

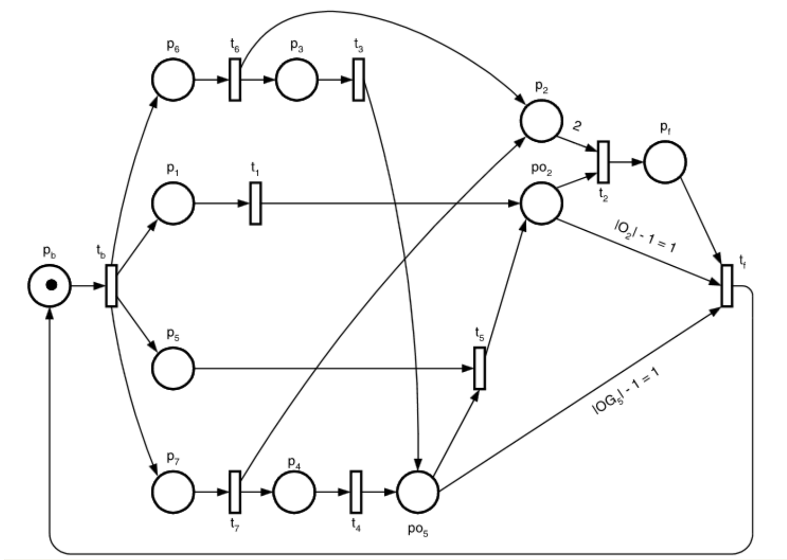


Figure 1.9 Disassembly Petri net (DNP) for sample product [Moo 01].

Note that Petri nets are often constructed when one needs to simulate systems that not only encompass sequences generation, but also include higher aggregated levels, such as task planning and extended process chain for more precise and successful analysis.

- Genetic algorithms (GA) approach is often chosen to solve optimization problems due to its capability in solving large and complex models compared with the other heuristic methods. GA being of stochastic type, the generation of optimized sequences is obtained using tailored fitness functions which consider the geometrical constraints of the product, the minimization of disassembly time and cost, as well as the possibility of grouping assembly operation or the environmental impacts [Yi 07]. A fitness function of the GA depending on the increment in disassembly time is present by Go *et al*. in [Go 12]. GA based approaches for disassembly sequencing of End Of Life (EOL) products were proposed in [Kon 06b, Giu 07]. A heuristic GA was developed in [Ric 13] in order to optimize partial disassembly sequences based on disassembly operation costs, recovery reprocessing costs, revenues and environmental impacts. Chen et al. proposed in [Che 01] a stochastic planning of assembly sequences using a two levels GA optimization where the chromosomes are the disassembly sequences which may be valid or not. GA Level 1 is used to generate an optimal assembly sequence using the current settings of GOPS corresponding to basic operators (crossover, mutation, selection). At Level 2 these probabilities are optimized by a second GA to generate new populations. The scheduling algorithm takes into account the geometrical constraints thus optimizing the physical constraints. The individuals (disassembly sequences) of each population have to be approved as to their trajectories.

- Fuzzy logic being dealing with uncertainties, an example of Fuzzy Logic–Genetic Algorithm methodology for automatic planning of assembly and disassembly sequences of products is treated in Galantucci *et al.* [Gal 04]. The main goal is to find the optimal sequence requiring the minimum completion time, by taking into account the fuzzy model of the processes and the constraints in available tool, destruction models, etc.

- Neural Networks viewed the disassembly sequence problem as a variant of the traveling salesman problem (TSP), which is to find a traveling sequence with the shortest distance to visits all the cities of the problem only once. A neural network consists of a large number of processing elements or neurons and weighted connection among them. Thanks to the highly interconnected neurons, a neural network can perform rapid computations in parallel and solve the computation efficiently. In [Hua 00] Huang presented a deterministic method for the economic analysis of disassembly process based on a method for the generation of disassembly sequences using a neural network. The later determines the best sequences of dimension *n* set by the user. As there is not 3D representation of components associated with this process, contact conditions, mobility and components' accessibility are not related to component models. Thus, there are no guarantees that the obtained sequences are valid. The optimization function used is similar to the traveling salesman problem by adding precedence constraints specific to the problem of disassembly. The approach described is subjective in nature and has no link with the 3D digital model of the product. It contributes to assess the best disassembly sequences from the viewpoint of recycling material.

- Note that the main disadvantages of Neural Networks are due to its lack of global searching capability and sensibility on the selection of the network parameters value and initial conditions. Bayesian networks are graphical models developed in the field of artificial intelligence as a framework that should assist researchers and practitioners in applying the theory of probability to inference problems of more substantive size and, thus, to more realistic and practical problems [Tar 13]. They integrate perception and action and use the dependencies among various parts of a product to propagate uncertainty regarding their condition as sensed during the disassembly process [Gei 96].

- Ant colony algorithm has been inspired and consequently developed from the observation of the operating mechanism of food exploration in ant colonies. Indeed, despite their limited intelligence ants collectively manage to find the shortest path to a resource of food: the most borrowed path is always the shorter one. Thus, Tripathi et al. [Tri 09] proposed an ant colony algorithm based on a probabilistic model between each step (of the algorithm), that can facilitate the choice of paths most likely to be the ideal disassembly sequence. Authors claim that this model provides faster and more accurate results as those obtained by a genetic algorithm.

1.5 Assessment

The bibliographical analysis shows that the existing methods for disassembly sequences generation developed so far, as we are aware, satisfy only partially the needs of designers end industrials. The majority of work contributes in one way or another in the modeling of these operations often based on simplifying assumptions and hypothesizes.

Thus, almost all simulation methods are based primarily on translational movements in order to separate or extract / insert components without rotation even if a numerical model is used to automatically identify the mobility of the components. Most often, this restriction is justified by the reduction in the associated computational complexity, even if it is at the expense of feasible solutions that are not identified.

Two large approaches for disassembly sequences generation based on the components' mobility modeling emerge, namely stochastic and determinist:

Most of the stochastic methods allow reduction of the combinatorial of disassembly operations thus defining a sequence. One of their major disadvantages is the impossibility of obtaining the same results when the input data is the same which does not allow to easily evaluating the influence of some input parameter on the simulation results. Some of these methods are limited by the combinatorial sequencing because they are based on geometrical information only. In one hand it generates a large number of possible sequences, then using technological criteria, contacts, etc, and in another hand a reduction of the number of solutions must be performed.

Concerning the deterministic interactive methods for sequence generation they are often subject to the discretion of the user. The later must simultaneously control the design process and the DFA approach as well. These methods reduce the combinatorial sequences but add the need to interactively specify a large number of parameters especially for complex products where many designers are involved. Waves propagation disassembly uses modeling component with reduced mobility. Only translations using the intersection of translational directions attached to each side of the contact surfaces to locally validate or not a disassembly operation. The Neural Networks approach described in [Hua 00] is also of subjective nature and has no link with the 3D digital model of the product. However, it allows assessing the best disassembly sequences from materials recycling viewpoint.

1.6 Summary synthesis and critical analysis

According to the applied techniques three large groups of methods for the disassembly sequences generation emerge, namely: *interactive*, *automatic* and *Artificial Intelligence* (AI) methods.

The review of current approaches, briefly presented here and other works we have studied, leads to the following remarks:

- Interactive methods (Section 1.3.1) being classified as indirect approach in the sequences generation for complex assembly, instead of gathering information from geometry, these methods get information directly from the user. They allow generating reasonable sequences for disassembly operation. However, the problem is that the disassembly operator (user) is not always the designer of the product. Consequently he/she may have some difficulties to answer the questions related to the product design thus compromising the search process.

- As mentioned in Section 1.3.2, automatic methods are preferred way for sequences generation. There will be no better way than generating the disassembly sequences automatically according to the geometry relationship among the components in a product. However, the ideal way is always the difficult way for sequences generation. How to find the easiest and simplest model for sequences generation in automatic way is still an issue.

- Searching algorithms: The wave propagation algorithm has a computational complexity of *O(sn2)*, Dijklstra's algorithm method proposed by [Gar 04] has the computational complexity of *O(nlog n)*, when $s<<n$, and *O(sn)* time when $s \cong n$.

- *AI* methods (section 1.4.4) are different of mathematical programming techniques, which are inspired by sophisticated methods thus allowing generating the possible disassembly sequences for a given product. They try to find the optimized way for sequences generation as well. However, they are always related with the expert system or online calculations for real application in industries and need relatively long execution times for sequence searching as mentioned in [Lam 03].

1.7 Objectives of the thesis

After the analysis highlighting the need for appropriate method for reducing the complexity for product disassembly sequences generation this section presents the objectives of the thesis. Given the assessment, the summary synthesis and the critical analysis presented here above our aim is to establish a simplified model for selective disassembly sequence generation. As it was mentioned in the General Introduction, when disassembling, it is important to eliminate the components which are unrelated to the target components prior to sequence generation. In order to address this configuration, our aim is to propose a method for generating the feasible disassembly sequences for selective disassembly based on the lowest levels of a disassembly product graph. Our goal is also to generate the minimum number of possible disassembly sequences. Thus, the proposed method has to include optimization aspect as well. Thanks to the proposed disassembly product graph (see Chapter 3), the method allows generating the minimum number of possible disassembly sequences. For this purpose prior to sequence generation, all the unrelated components with the disassembly target(s) are eliminated from the graph and the process automatically stops when the target(s) is (are) reached thus reducing computational research time.

We present also a set of support modeling software (see Chapter 4) for selective disassembly sequences generation thus allowing assisting designers in their work in the initial stage of a product design. This software is currently being integrated within the framework of a "trade application" to EUCLID-V5 Software.

1.8 Conclusion

In this introductive chapter, first the synthesis of the most recent literature survey, to our knowledge, of Disassembly sequencing was done. Note that the common point of all these methods is that they are often a combination of different methods for Disassembly sequences generation and methods for optimization. The principal common difficulties of all these methods are their limitations in motion presentation as often the translations only in some general directions are considered and the rotations are not taken into account.

Automatic disassembly sequencing is an ideal way for the disassembly sequencing. However, there are two major problems in this field. One is the representing models for the product. There are many graphs or networks as presented above for representing the relationships among the parts in a product. However, the graph-based techniques for example, do not consider products geometrical information data bases. As we are aware, there are not works mentioning that these graphs or networks can be built automatically according to parts‘ relationship in the product. The other problem is related with the calculation method. Basically, all the graphs can be translated into matrix calculation for sequences searching (or calculation). Many works, based on some simplifications hypotheses, only focus on four or six directions to disassembly the product which is easy to transfer the disassembly calculation into the matrix computation model. However, in the real situation, the components disassembly direction of translation cannot be just in four or six directions. The motions of rotations have to be taken into account as well.

Searching algorithms allow reducing the computation resource for complex models. However, the 3D assembly corresponding graphs are often difficult to build. Thus, these algorithms are relatively difficult for real application in product disassembly. Most of them involve simplification in the assembly model of the product in order to avoid complex computation resources.

Regarding the AI methods they do not only focus on the disassembly sequencing of the product. As previously said, they concern the optimization of the sequences as well. In most works, the sequences already exist and AI method tries to choose the best one under some constraint.

In order to reduce the effect of these simplifications and to obtain a proper disassembly sequence planning, a complete removal model is required, which shows the importance of model processing for disassembly simulations. Thus a more general method is necessary including the movements of translation, rotation and the collision detection as well.

On the other hand, this chapter aimed to provide arguments concerning additional scientific developments in order to propose new more suitable and effective models and methods than the existing ones. Thus, these elements represent an introduction to the following chapters which will focus more on the detailed description of the proposed models.

Chapter 2

Virtual reality for assembly/disassembly operation simulation

This chapter analyzes the results and some remaining problems of existing research in the field of Virtual Reality (VR) technology related with disassembly operations simulations. First an overview of the existing VR integration approaches is presented by insisting on the constrain-based and the physical-based. The Haptic interaction and force feedback are also presented. Then some Virtual assembly platforms which incorporate Ergonomics analysis are presented, followed by a Critical analysis and assessment, thus allowing introducing the Proposed VR environment for Disassembly operation simulation: sequences generation and their evaluation.

2.1 Introduction

In the today's global context, virtual manufacturing become critical to increase the efficiency of Product Development Process (PDP). This manufacturing involves computer graphics and simulation methods to model the real world. Assembly and disassembly (A/D) operations are key components in many manufacturing processes. Consequently they should be also supported in a virtual manufacturing system as a realistic A/D process modeling can reduce cost and save time.

Virtual reality (VR), is defined as "*a scientific and technical domain exploiting the possibilities of computers and behavioral interfaces to simulate in a virtual world the behaviors of 3D entities, which interact in real time with each other and one or more users in pseudo-natural immersion through sensorimotor channels*" [Fuc 06]. VR technologies provide advanced methods of real time user interaction. Their realistic behaviors of animated bodies enhance the feeling of immersion and improve performance of the user. Today VR environments have significantly evolved towards A/D simulation, highlighting new needs for A/D simulation preparation, evaluation and their integration in PDP. In order to save time and improve PDP, many works focus on virtual reality simulation [Duv 13] and in particular on A/D process [Jun 03, Gar 07, Ash 09, Wan 06, Li 12, Pon 13a, Pon 13b, Iac 14]. All these simulations address different objectives such as: A/D sequencing, path planning, collision detection, operational time evaluation etc., which often are complementary to each other. Thus, VR is a new technology that creates a real-time visual/audio/haptic experience with computer systems. It provides a potential way for disassembly operation simulation.

In this context, recent work and techniques related with A/D operation simulations and evaluations in VR environment are presented in this chapter. Its goal is to assess the advantages and the shortcoming of this technology, and the remaining problems of the existing works in order to present our contribution in this field.

2.2 VR integration approaches overview

As mentioned here above, VR technology plays a vital role in simulating advanced *3D* human-computer interaction by providing users with different kinds of sensations (visual, auditory, haptic, …). Virtual assembly simulations allow designer to evaluate the concepts in virtual environments during the early design stages. With virtual prototyping applications, for instance, the optimizing process for the design for assembly can be incorporated in the conceptual design stage. Using haptic or auditory technology, allows designers to interact with the assembly parts through human basic motions. Thus, contact force may be transmitted to the operator in real time.

In 2003 Kim and Vance [Kim 03] described Virtual assembly as "*ability to assemble CAD models of parts using a three dimensional immersive user interface and natural human motion*". Later, Seth *et al*. [Set 11] defined virtual assembly as "*the ability to assemble virtual representations of physical models through simulation realistic environment behavior and part interaction to reduce the need for physical assembly prototyping resulting in the ability to make more encompassing design/assembly decisions in an immersive computer-generated environment*". The applications scope of virtual assembly is large as shown in figure 2.1.

Figure 2.1 Applications of a virtual assembly/disassembly simulation [Set 11]

Let us remember, that VR systems for assembly can be roughly divided into two categories. The first one is the virtual assembly systems using relative technologies to improve the feeling of the operator. The other one is virtual assembly systems for immersive assembly modelers which additionally support the combination of CAD-based parts to novel assemblies. The first category purpose is to check the general assemblability of the design: part accessibility, tool usage, generation of sequences and trajectories of assembly operations, and so on. The second category focuses on some sort of snapping mechanism which automatically completes an assembly operation when two parts close enough (in the vicinity of the contact) are moved in a virtual environment.

In the most of today's industrial applications, VR systems are still used as mere visualization tool for prototypes modeled in external CAD systems. It not only provides powerful modeling functionalities but also, in particular, can be easily instructed, e.g. by meanings of intuitive language and gesture-based instructions. Figure 2.2 shows, an example of virtual environments involving the assembly of CAD-based parts [Jun 03].

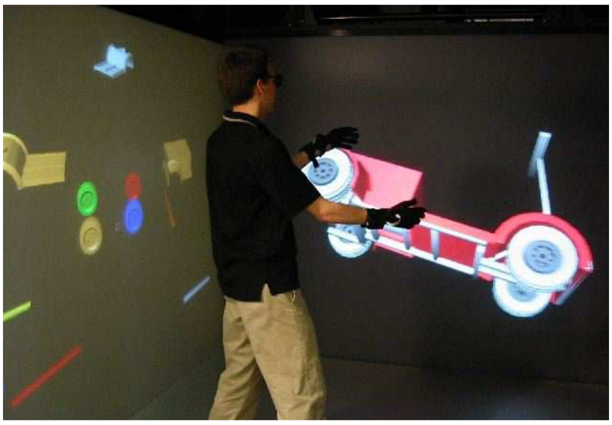


Figure 2.2 CAD-based assembly in Virtual environment [Jun 03].

Note, that there are two kinds of systems for virtual environment development. One is constrain-based modeling approaches system the other is physical- based system**.**

2.2.1 The constraint-based

Constraint based modeling approaches are using inter-part geometric constrains to determine the relationships among components in the assembly. It consists of writing each contact as a unilateral constraint and solving the resulting system of equations for the object positions. Compared to the physical based simulation (see Section 2.2.2), constrain based modeling has two advantages: less computationally intensive and available information in the CAD models [Mar 03]. Constrain based application can produce much more realistic results, without unwanted artifacts and with the possibility to computer contact friction correctly [Per 13, Tch 10].

There are two types of constraints modeling, which are positional constraints and geometric constraints [Set 11]. Position constrain can be represented by a set of equations, which can be solved based on numeric, symbolic or graph based method [Gao 98]. Instead of translation position constraints into equations, a geometric constraint focuses on the rigid body transformations which satisfy a set of constraints presenting the prelateships among all components [Leu 13].

For the constraints based application, there are many systems developed which can support the constraints detection. For assembly planning a non-immersive desktop and immersive CAVE environment were proposed by Cruz-Naira *et al.* in [Cru 92, Cru 93]. It provided the subjects with a more immersive sense of virtual assemblies by implementing with IRIS performer. The results from the immersive VR environments, non-immersive VR environment and traditional engineering environment were compared by the authors, which showed that the subjects, in virtual environments had better performance than the performance in the traditional engineering environments.

Luis Marcelino [Mar 03] implemented a geometric constraint manager designed to support physical realism and interactive A/D tasks within virtual environments. The key techniques for this application are direct interaction, automatic constraint recognition, constraint satisfaction and constrained motion. He described the development and the implementation of a Geometric Constraint Manager which is used in Real-time immersive virtual environments such as CAVE. The system architecture of the constraint manager includes *constraint solver* and *constraint recognizer*. The *solver* determines the transformations to be applied to unfixed objects. It also applies new constraints, removes existing ones and fixes objects in 3D environment. The *recognizer* focuses on identifying new possible constraints and validates existing ones. However, the constraint manager has two bottlenecks. One is the recognition process. The other is the transformation of objects, which results in the unacceptably low frame rate.

Jayaram *et al.* [Jay 99, Jay 06, Jay 07] developed VADE (Virtual Assembly Design Environment) system for assembly simulation (Fig.2.3), which can support constrained motion simulations. The simple constraints such as: against, coincident, etc. can be detected automatically and the relative motions of the objects can be based on the available constraints.

The system imports all the data information (transformation matrices, geometric constrain, assembly hierarchy etc.) from CAD model to perform the assembly simulation. In 2001, a physics-based algorithm was added to VADE by Wang *et al.* in order to perform more realistic part behavior [Wan 01]. The VADE system has been used for virtual assembly simulation in various applications some of which are presented in [Jay 07]. A CAD linked virtual assembly and maintenance simulation based on constraint-based modeling was presented by Wang *et al.* in [Wan 06]. A year later, Yan *et al.* [Yan 07] developed a system for assembly path planning based on the constraint-based modeling. However, as the number of contact points is large, the constraint-based methods are much more difficult to implement.

In fact they take a lot of time for computing and the results may not always be stable as pointed out by Leu *et al.* in [Leu 13].

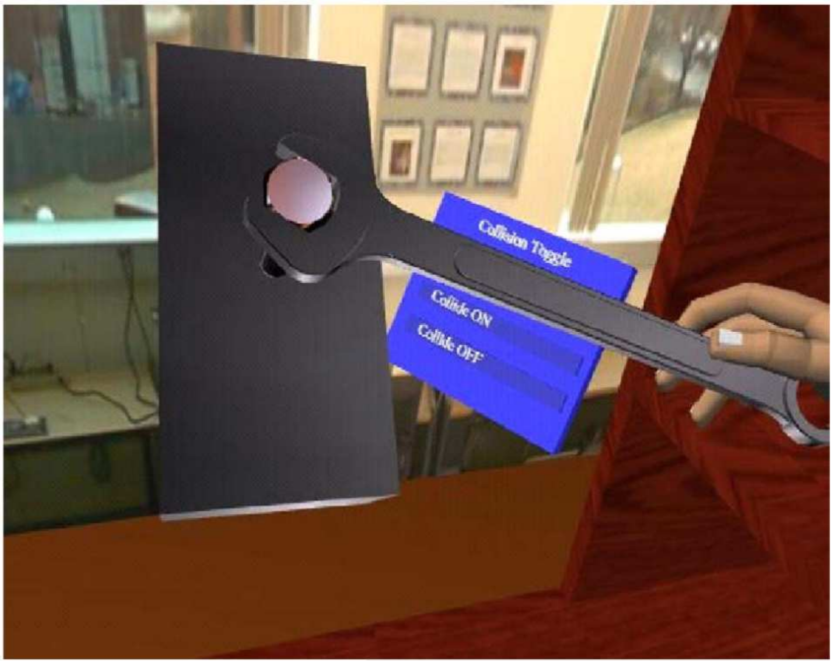


Figure 2.3 VADE environment [Jay 06].

2.2.2 Physics-based system

Physics-based modeling simulates realistic behavior of parts in virtual scene, where parts are assembled with each other. This modeling is based on the simulation of physical interaction in time. It is applied primarily in the interactive dynamic simulations with human operators involved. The method always accurate and fast collision detection thanks to the calculated velocities and forces at the contact points. The forces can be returned to the operator through force feedback devices. Note, that there are two Physic-based types modeling algorithms based on the used method. One is the Penalty force, the other is Impulse method [Leu 13].

- Penalty force method

  Penalty force uses a spring damper system for preventing interpenetration between models. Using Hooke' law the penalty force is:

  $F=-kdN$,

  where k is the object stiffness, d is the shortest distance from the tool point to the object' surface, and N the vector from tool point to the contact point. In the same time, the force value is affected by the elasticity of the object. For hard collision, k

will be the same whether the objects are approaching or receding. The penalty force methods are easily for implementation. However, if the spring stiffness is too high, the stiff equation is numerically intractable [Set 11]. A physics-based virtual assembly system was presented by Garbaya *et al.* in [Gar 07] (Fig. 2.4). The approach, mainly based on spring-damper model, focuses on part to part interfacing and contact force during mating phase of the parts assembly in the VE. The collision detection system is based on V_CLIP algorithm. In this approach, each model is presented by two 3D models: tracked model and visual one. The Tracked model is created by VHT (Virtual Hand software toolkit) library. The visual model is created in PhysX engine, visualized by OpenGL render. During the mating phase of assembly operation, the spring-damper model is used to render realistic parts behavior and contact force sensation. The study concludes that the user performance increased when inter-part collision forces were rendered to him/her. Some other related works can be found in [Erl 05, Ale 11].

Figure 2.4 Data glove in a six-sided CAVE. [Gar 07]

- Impulse method

The interaction among objects uses collision impulses for all types of contacts. The impulse from collision is calculated to find out the absolute velocity of an object at the contact point. In this approach, the static contacts are considered as the high-frequency collision impulses. The method is more stable and robust compared with penalty force method. According to Seth et al. [Set 11] and Leu et al. [Leu 13] it cannot handle simultaneous and stable contacts like sliding and stack of blocks at rest. However, Renouf et al. [Ren 05] disagreed with them.

2.3 Virtual assembly platforms

In last decades, different virtual assembly platforms using different assembly techniques have been proposed by many authors. Virtual reality platforms can be used to simulate the whole manufacturing process including the assembly and disassembly operations. All this platforms are physics-based, constrains-based or a combination of both of them. Germanico *et al.* [Ger 13] presented a literature review of different VR platforms as shown in Table 2.1.

Table 2.1 Key features of some virtual assembly platforms [Ger 13].

System	Year	Assembly method	Key Features	Haptic Device
HIDRA[16]	2001	Collision detection	Integrates a haptic feedback into a (dis) assembly simulation environment Manipulate parts using two fingers	Phantom desktop
MIVAS [10]	2004	Physics based	Optimization techniques for complex models and assembly operations Tracking of user movements and voice commands Realistic virtual hand interaction for grasping of virtual parts Documentation of assembly planning results	CyberGrasp one hand
VADE [12]	2004	Physics and constraints	Users can perform the assembly using hands and tools such as screw drivers During the assembly process VADE maintains a link with the CAD system Let the user to make decision and design changes Swept volume generation and trajectory editing	CyberGlove two handed
SHARP [11]	2006	Physics based	Capability of create subassemblies Swept volumes for maintainability Network module for communication with different VR systems Portable, runs on different VR systems such as HMD, CAVE, projection walls and monitors	Phantom omni dual handed
VEDAP-II [7]	2009	Physics based	Oriented to assembly planning and evaluation Focuses on modeling the dynamic behavior of parts during virtual assembly operation	CyberGrasp one hand
MRA [21]	2009	Physics based	Use of low cost technologies for two hands assembly Real scale projection and tracking system to change point of view The system demonstrates the assembly procedures and the user must repeat it	6D35-45 / Wii-mote
VCG [3]	2010	Constraint based	Oriented to assembly planning and training Method of constraint guidance to perform the assembly Use of virtual fixtures, use of mechanical constraints, intuitive assembly, on-line activation of kinematic constraints	Virtuose 6D35-45 one hand
IMA-VR [5]	2010	Constraint based	Virtual training system for the cognitive and motor skills transfer combining haptic, gestures and visual feedback Uses the concept of spring-damper model to avoid parts interpenetration Visual dynamic behavior of parts to represent manipulation of real parts	Phantom/ LHIfAM/ GRAB two hands
HITsphere system [6]	2011	Physics and constraints	Immersive virtual environment with walking capability to simulate ground walking Free manipulation of virtual objects Automatic data integration interface Constraint-based data model is rebuilt to construct the virtual assembly environment	Phantom Premium one hand

2.4 Ergonomics analysis in VR system

In recent years Ergonomic assessment of manufacturing industry in VR system is becoming more globalized [Wil 99, Jay 06, Pon 13c, Pon 13d]. Ergonomic engineering deals with human behavior capabilities and their limitations in workplace which have to be taken into account during the design of a product or a system. The philosophy of such engineering is to fit the task to the human and not the human to the task. The key point for an effective application is to gain a balance between the human body characters and the task demands. Various commercial software systems are available today for ergonomic studies in general and for their evaluation in particular. The Car interiors evaluation for example the JACK, ANNIE-Ergo man, and RAMSIS, amongst others, have good performance [Wil 99].

In virtual reality, Jack's Task Analysis Toolkit (TAT) for instance (Fig 2.5) is a tool for ergonomic analysis of virtual human movement. It allows analyzing the posture of Jack in order to detect and consequently resolve ergonomic issues.

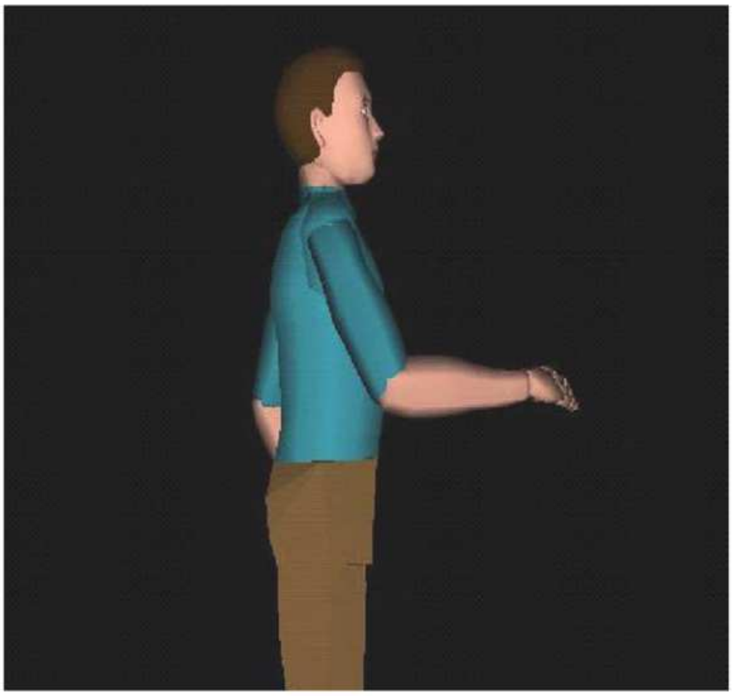


Figure 2.5 Jack ergonomic analysis tool [Jay 06].

Rapid Upper Limb Assessment (RULA) algorithm [McA 93] for ergonomic evaluation allows assessing the risk of upper limb disorders. It is based on some parameters such as: postures, muscles use and weight of loads. The data required by RULA worksheet includes: joints angles and twisting of the arms, wrist, neck, trunk and legs, as shown in Fig. 2.6.

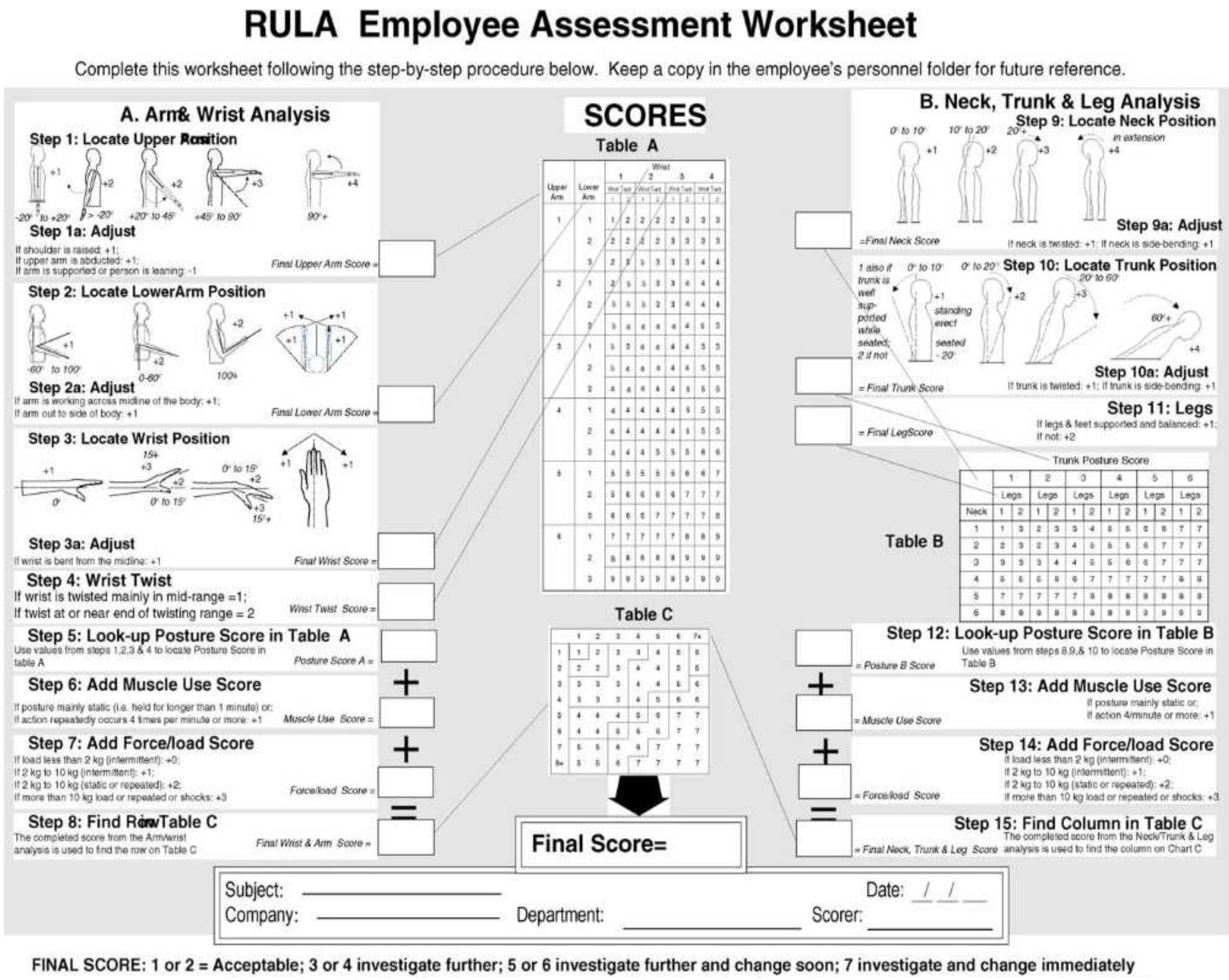


Figure 2.6 RULA Sheet [McA 93]

However, there is still a low level of acceptance and limited application of ergonomic analysis in the manufacturing industries. The main problem is that ergonomic analysis always involves a *3D* human model to replace the real human for the ergonomic condition simulation. There are two strong limits in this field. The first is leaving the human aspect out of the assembly planning which could result in incorrect or inefficient operations. The other one is that as the number of parts is increasing exponentially for complex products, consequently it becomes more difficult to characterize criteria for choosing the most suitable assembly sequence for a given product. Task performing is becoming too complicated for the 3D human manipulation in the VE, that it needs involving high investment. Therefore, according to Atsuko *et al.* [Ats 13] due to the high cost investment, human model application areas are applied mostly to mass and highly cost product industries such as the automotive and aircraft industries.

2.5 Haptic interaction and force feed back

Nowadays, with the commercialization of new haptic technologies and software development platforms, the simulation of force feedback in virtual environment applications started to become more widespread. A haptic device, also called feedback device, is a computer peripheral which provides the force to the hand of its user. The input information is the movement of the haptic device (called also effector), and the output is the force. The behavior of the object is controlled by the physics' laws. (Rq. Here, the notion of "object" is used in the most general case, while the notion of "component" is used for the parts of an assembly).

Haptic technology can be divided into two categories depending of the wearing way which are: non-portable haptic device and wearable haptic gloves.

- Non portable haptic devices:

  Sensable Technologies PHANToM (http://geomagic.com/en/products-landing-pages/sensable),

  Immersion's CyberForce (http://www.immersion.com/),

  Haption Virtuose (http://www.haption.com/),

  Novint Falcon (http://www.novintfalcon.com/).

- Wearable haptic devices:

  CyberTourTM, CyberGraspTM (http://www.immersion.com/),

  Rutgers Master II [Bou 02].

In certain fields, such as the training skills for surgeons, it has been shown that the haptic feedback improved dramatically the performance [Pop 00, Sal 97, Ric 95]. The use of haptic interface to feel collisions of 3D models in assembly tasks is presented by Ladeveze et al. in [Lad 10]. Haptic provides repulsive forces allowing preventing motion for the hand and realizing the path planning (Fig 2.7). However, the device has limitation of workspace which restricts the movement of the operator in the environment.

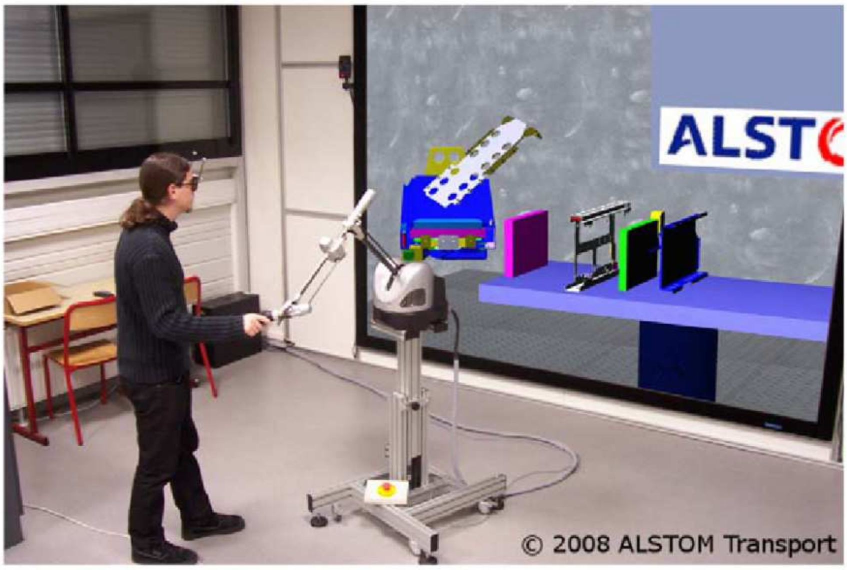


Figure 2.7 Assembly simulation using haptic device [Lad 10].

The question is whether the cost and associated effort of integrating the current haptic devices into a computer-generated simulation are worthwhile. Thus, for industrial assembly tasks, for instance, the question is whether the operator could benefit from the haptic feedback or not. In this context Edwards et al. [Edw 04] investigate whether auditory cues would be as effective as force feedback cue for an assembly task and whether subjective evaluations of usability would differ as a function of the type of feedback information provided to the user. For this purpose 24 volunteers (males and females) were involved to assemble and disassemble five interconnecting virtual parts with either auditory, force, or no feedback cues provided. The performance for the task was measured by completion time and the number of collisions between parts. Note that some of the factors that make difficult incorporating force feedback into virtual Environments are: high cost for devices with higher degrees of freedom, limited amount of fidelity provided with current haptic devices, large processing requirements and slow update rates of current devices. The last two factors are the causes that most of the devices can only provide the main types of haptic feedback, either kinesthetic or tactile sensations. The main objectives focused by the authors are divided in four items. The first one is to determine whether the force feedback increases performance and eases the interaction for a simulated assembly task performed within an immersive Virtual Environment (VE). The second item is to determine whether force feedback cues (selections) made the interaction more realistic for the users and potentially allow them to do their task more proficiently. The third one is to determine whether auditory cues provide an effective substitute for force feedback. And finally, the fourth item is to establish whether having both auditory and force feedback cues would increase performance and subjective measures of user satisfaction.

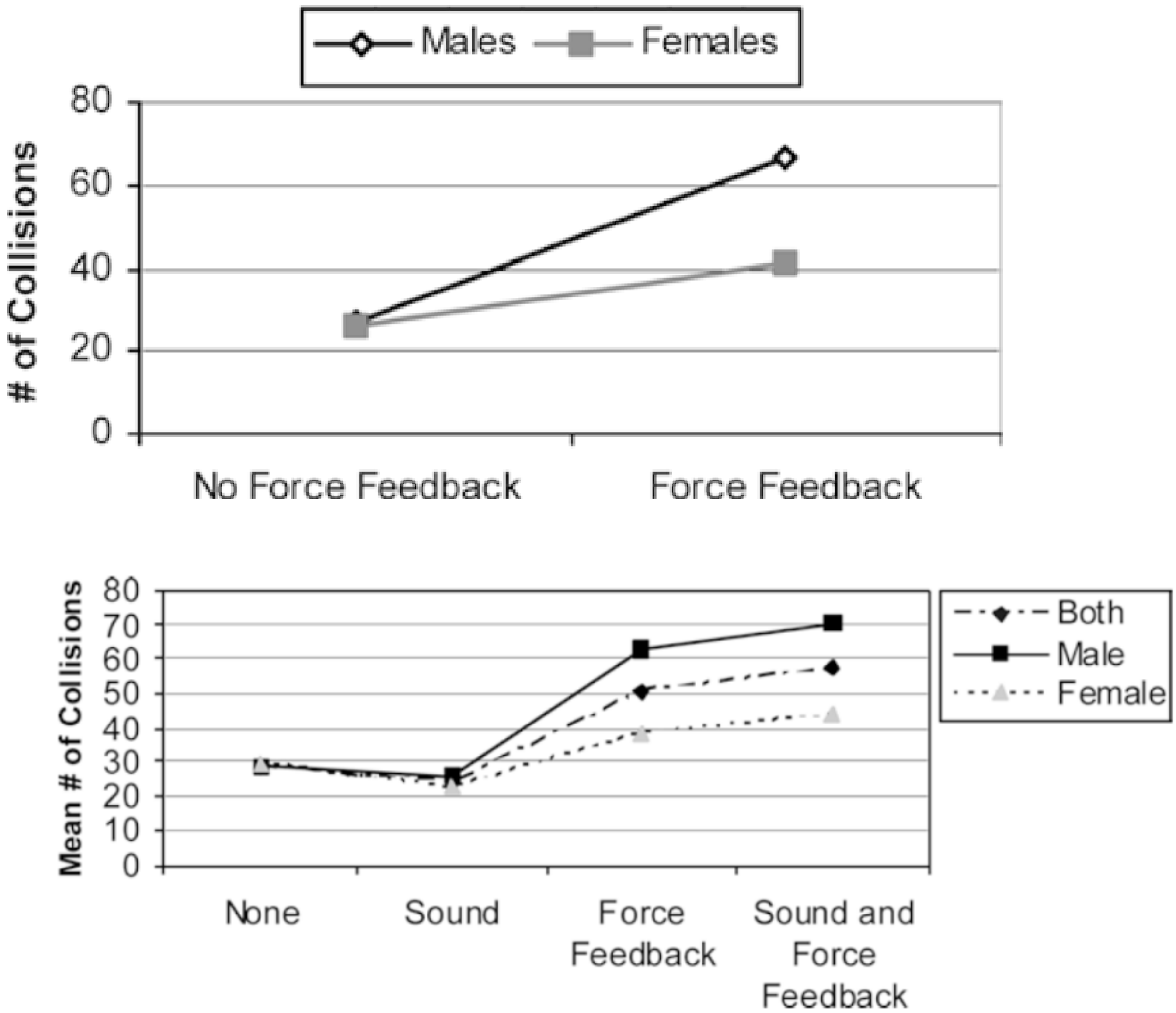


Figure 2.8 The mean number of collisions by condition and force feedback effect [Edw 04]

The results of this study (Fig. 2.8) show that the addition of force feedback actually decreased performance. The possible explanation is that the simulation is not realistic enough to benefit from force feedback. As for the second objective, the study found that male's volunteers reported an increased sense of realism with the addition of force feedback, while females only showed a tendency to rank force feedback as more realistic. The study shows also the auditory cues neither increased or decreased completion time or the number or collisions. The sound may be an effective means of conveying collision information, since sound did not negatively affect manipulation performance while still slightly increasing use-perceived realism and overall satisfaction. The results indicated also that participants rated the combination of sound and force feedback higher than all other conditions for perceived usefulness towards a real assembly task. As the results shown, the effect is not obviously as expected, and the reason maybe the level of simulation. Level means the simulation precision of the involved sound and force feedback. In fact, authors forgot one thing that may affect the whole results: the precision of calculation for each simulation, especially for the feedback cues and the auditory.

In all, haptic technology cannot improve the performance in the absence of high precision simulation. The Non portable haptic devices have limited workspace for interaction. The Wearable haptic devices, however, provide force feedback only to fingers and palm, which is limiting their application field.

2.6 Critical analysis and assessment

Most of the recent works on A/D related with VR technology focus on the simulation itself. They try to build an environment to assembly or disassembly products and to compare the simulation results with the results of real A/D process. Some works use VR as the A/D path planning [Ale 11]. Some commercial software tools were also proposed to perform ergonomic evaluation during assembly [Jay 06, Set 11]. However, for virtual A/D simulation, several challenges need to be overcome.

- Collision detection: Collision detection prevents part interpenetration during virtual simulation. The fast and robust 3D collision detection algorithms are always required in the applications of Computer Graphics. Foisy and Hayward [Foi 93] presented four groups of algorithms which are: space-time intersection, swept volume interference, multiple interference detection and trajectory parameterization. In our opinion, the so called Extrusion operation proposed by Cameron in [Cam 90] is the most general representation of collision detection problem. The collision between two objects occurs if, and only if, their extruded volumes intersect. The extruded volumes approach is partially occluded by the high cost of its practical implementation, whose bottleneck is the generation of the 3D extruded volumes themselves.

  For this reason, other approaches have been proposed aiming at avoiding this explicit computation. These approaches can be divided into two groups, namely: geometric and algebraic.

For the geometric group approaches, two main alternatives have been proposed: projecting the extruded volume onto a lower-dimensional subspace leading to the swept volume approach and sampling along the trajectory. For the swept volume approach, the volume containing all the points occupied by a moving object during a time period is called the swept volume (see details in Chapter 3). If the swept volumes for the objects in a scene do not intersect, there is no collision happening. However, the generation of swept volume is too computationally expensive. Thus, many works adhering to this approach deal with convex approximation swept of volume instead. For example the object trajectories, concerning the multiple interference detection, the aim are to sample the object trajectories and repeatedly apply a static interference test. Of course, the sampling way is crucial for the success of the approach. The ideal way is that the next sample should be the earliest time at which the collision can really occur.

Concerning the algebraic field approaches, the aim is to parameterize the trajectory. Thus, the approach called trajectory parameterization focuses on the objects trajectories functions depending of a parameter (time). However, depending on the trajectories, the degrees of the resulting polynomials may be arbitrarily high, and the determination of the collision instant can be computationally very expensive for arbitrary trajectories.

Note, that the key issue here is the cost of the collision detection. Thus, the right time and place to apply the detection test become a key aspect of any polyhedral collision detection scheme. It is possible to bound the time interval when the collision occur if we know the objects' moving direction and how far away they are from one another. In [Jim 01] authors mentioned that most of collision detection schemes only deal with polyhedral approximations. However, this is a great challenging problem for collision detection not only due to the nature of the objects' motion but also because, in this case, a polyhedral approximation is inadequate.

- Constraint based or physical based models: As mention before, no matter constraint based or physical based model for inter-part detection, there are great challenges for both of them. The constraint based (position or geometric) modeling always needs the predefined geometric constraints before the implantation in virtual simulation. Then the system can compute relative motion of the objects based on available constraints. Compared with physic-based modeling, constraint based method can reduce computing time when a lots of components are involved in the scene. However, the predefined task will be heavy. If the task is related with non-linear systems, constraint based method still have difficulties handling over-constrained situation. For complex constraints simulation, exhaustive computation requirement make it inappropriate for real world application.

Concerning the physics-based modeling to simulate realistic behavior among complex parts interactively and accurately is still a challenging task. Compared with constraint base method, this method needs more computation time, especially, when several contacts occur simultaneously. It can get more accurate force feedback results. In most works, this method focuses on simple two parts assembly simulation.

Therefore, choosing a better algorithm for inter-part detection is still a great challenge at the present as well.

- Evaluation methods: How to evaluate the disassembly or assembly is the main concern of our thesis. Most of the recent works on A/D related with VR technology focus on the simulation itself. They try to build an environment to assembly or disassembly products and to compare the simulation results with the results of real A/D process. As previously said, its purpose is to try to fit the task to the human and not the human to the task. To evaluate some ergonomics parameters during simulation in VR environment, most works focus on using a human model in a digital mock-up (DMU) model [Wan 12, Lon 06].

Thus, how to find a more realistic method for disassembly/assembly operation simulation and their evaluation in VR environment is still an issue.

2.7 Proposed VR environment for Disassembly simulation

When it is necessary to analyze the interaction between human and object, VR technology is a better choice. Especially for disassembly sequences evaluation there is a strong need to evaluate disassembly operation for immersive simulations with a larger set of possible movements and to get more realistic results. However, for a complex product, the number of the possible disassembly sequences may be relatively large. If all the sequences have to be evaluated in the virtual reality, it will be relatively costly on time. So, prior to evaluate, most of useless sequences need to be deleted.

Thus, we propose a VR environment for A/D simulation particularly suited for scientific and industrial applications. The component models are acquired through a STEP file coming from a CAD software (see details in Chapter 4).

2.8 Conclusion

Despite of their many advantages, the majority of the A/D methods related with VR technology, a part of which were described above, have certain shortcomings to make them convenient for industrial applications.

Most of the recent works on A/D related with VR technology focus on the simulation itself. They try to build an environment to assembly or disassembly products and to compare the simulation results with the results of real A/D process. Some commercial software and tools were also proposed to perform ergonomic evaluation during A/D. However, as mentioned before, this evaluation is relatively expensive and often used only by mass production industries. Moreover, VR-based applications use real-time interactions and immersive techniques allowing enlarging the user perception of digital models. For disassembly operation simulations, relative mobility between components becomes also a major issue to reduce the simulation time and improve the efficiency of digital models.

After reviewing some current approaches, a part of which was presented here above, it can be stated that the existing VR approaches still have limitations in evaluation of disassembly sequences complexity. So, there is a strong need to evaluate disassembly operation for immersive simulations with a larger set of possible movements and to get more realistic results. In addition, when haptic devices are used, penetrations due to collision detections can be avoided and the quality of the user's feedback can be improved. For this purpose component mobility need to be characterized to specify constraints associated to haptic devices during the A/D process simulation.

This chapter aimed to provide arguments concerning additional scientific developments in order to propose a new software tools more suitable and effective than the existing ones. These elements represent an introduction to Chapter 4 which will focus more on the detailed description of the proposed tool.

Chapter 3

Method for disassembly sequences generation

For disassembly it is important to eliminate the components unrelated to the target prior to sequence generation. In order to address this configuration, a method for generating the feasible disassembly sequences for selective disassembly is presented in this chapter. It is based on the lowest levels of a disassembly product graph. Instead of considering the geometric constrains for each pair of components, the proposed method considers the sets of disassembly for removal (SDR) and collisions among the components in order to generate the Disassembly Geometry Contacting Graph (DGCG). It is built according to proposed three micro units, which consider all the possible situations of relationships, among the components in the DGCG. The latter is after then used for disassembly sequence generation. The method allows reducing the number of possible disassembly sequence by ignoring the unrelated components with the target. It is applied for automatic generation of the selective disassembly sequences for different assemblies and is illustrated through two examples.

3.1 Introduction

Disassembly sequencing involves the search for all possible ways for disassembling and often selecting the optimum solution out of these. For the company, the improvement of the recyclability performance of their products is becoming an integral part in the product development process (PDP). Let us note that there are two major key concepts related with the disassembly applications. The first one is that considering the whole life of a product, designers have to integrate assembly and disassembly (A/D) operations in the earlier stage of a product design. The other concept is that today disassembly is based on the so called concept of "*selling use*" instead of *selling products*.

According to the assessment, the summary synthesis and the critical analysis presented in Chapter 1 almost all the methods related with the disassembly sequencing in the literature have some shortcomings. Thus, the aim of our thesis is to establish a simplified model for disassembly sequence generation. Let us remember that for interactive methods for disassembly sequencing generation, the operator is not always the designer, and often is not familiar with the structure of the product which led to the interruption of product design information. Thus, it will be great trouble for the operator to answer questionnaires about the product. For automatic methods, the graph based methods and Artificial Intelligence (AI) methods often focus on the theories development and have difficulties to consider complex products with multiple relationships among their components. Few other methods answer the question how to use the relationships in real product in order to automatically build the corresponding graph. In this chapter, a new method for selective disassembly based on the *Disassembly Geometry Contacting Graph* (DGCG) is presented. We claim that it is better than the method of DSSG proposed by [Smi 11, Smi 12] as it enables all the removing directions of the components to be taken into consideration. Instead of considering the disassembly direction, often limited in number of *4* or *6*, our method focuses on the s*ets of disassembly for removal* (SDR), including: both translation and rotation movements, and collision detection as well. The method is also better than the wave propagation method [Sri 00, Sri 99b] as its computation complexity $O(sm(n-1))$ is lower than the computation complexity $O(sm^2)$ of the wave propagation method thus reducing the number of search iterations to generate the possible disassembly sequences (m is the number of components in an assembly, s is the number of targets and n the number of level in the disassembly graph (see Sections 3.3)

Let note, that the method includes also optimization aspect as well because it converges to the minimum number of possible disassembly sequences. Thanks to the established DGCG graph, it allows generating the minimum number of possible disassembly sequences and rapidly converging to the solution. For this purpose prior to sequence generation, all the unrelated components with the target(s) are eliminated from the graph and the searching process automatically stops when the target(s) is (are) reached. The unrelated components are the set of components which remind in the assembly. They have not to be moved in order to reach the target(s) and consequently they do not appear in the graph. Our research is focused on selective non-destructive disassembling, rather than on destructive or complete disassembling (see Chapter1).

3.2 Relative concepts and definitions

Before presenting our methods, let us remember that disassembly models are often based on graph representation of the product where vertices represent parts, and edges constraints or contacts. Graphs can generally be converted into matrices for computation. In order to reduce the complexity of selective disassembly sequences generation, our approach uses the concept of the Gaussian sphere [Pom 04, Woo 91, Woo 94]. The related main issues for product disassembly are presented with some details in the following paragraphs.

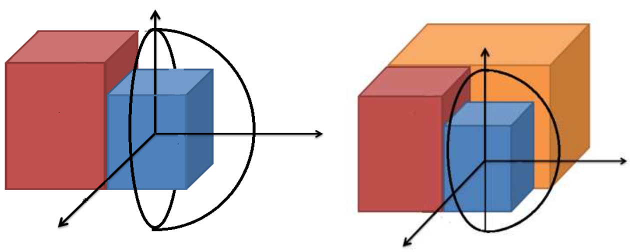
3.2.1 Contact identification

For contact identification the so called contact identification operator proposed in [Iac 08, Iac10] is used. The realized simulation framework Simpoly [Iac 08, Iac 10] automates the contact identification and offers a more robust approach for further usage of haptic devices. Thanks to the developed software, the operator can identify the different types of contacts in the assembly. The user has to select the components in the assembly tree built from the 3D model (STEP format) file where contact identification should be performed. If two components at least are selected, Simpoly generates a List of Bodies intersecting each other. To achieve this, the bounding box of each component is used in order to check the intersection between the bodies and speed up the process.

3.2.2 Set of directions for removal (SDR)

In order to represent all possible movements of translation and rotation for a part, we applied the set of directions for removal (SDR) approach proposed by [Sid 97]. SDR represents all the possible separation directions of a component with regards to the other components in the product. The basic idea of this approach is using the contact surfaces to determine the SDR. In [Woo 91] Woo and Dutta proposed an automated approache for disassembly path generation based on determining the pole vector of the Gauss unite hemisphere. Later the SDR method was modified by [Pom 04]. Authors described a method, based on the non-destructive disassembly plan, called modeling of assemblies, which allows determining the disassembly path and its optimization with adaptive planning after determining each movement of the components. The disassembly model is generated automatically from the CAD design of the product. The proposed model, not only facilitates the determination of the directions of removal for each component. It allows also capturing the necessary information for simulation of the disassembly process.

To remove the target component from a product, each contact needs to be checked first in order to get its possible SDR. If a component is free in the space, it can be moved by translation, rotation and helical motions in any direction. So, for a free component, its SDR in Gaussian sphere presentation will be the full sphere. If a contact between two components exists, *Plan Fit* contact for instance (see Fig. 3.1a), the contact surface restricts the directions along which the component may be removed, thus reducing the Gaussian sphere to a half. Another example with two *Plan Fit* contacts reduces the Gaussian sphere to a quarter, as shown in Fig.3.1b. Note that SDR concept concerns both components and contacts.



(a) (b)

Figure 3.1 SDR using Gaussian sphere based method.

From Fig. 3.1(a) and (b), it is seen that there are one and two mating faces respectively. If considering a sphere of unit radius (Gauss unite sphere), a mating face divides the sphere into two hemispheres. The hemisphere, labeled Hi, which corresponds to the outward pointing unit normal of the mating face, conserves the sets of direction for removal (SDR).

$$SDR = \prod_{i=1}^{N} H_i \tag{3-1}$$

where N is the number of the mating faces for the components. If the foot of the normal to the *i*-

th mating face is the pole, P_i (x_i, y_i, z_i) of the hemisphere H_i, then:

$$x_i x + y_i y + z_i z \geq 0 \tag{3-2}$$

and the analytical equation of the hemisphere H_i, containing all the sets of direction for removal is:

$$x^2+y^2+z^2=1 \tag{3-3}$$

Therefore, for the SDR, according the eq. (2-3) and (3-3), the function (1-3) can be calculated from the intersection of these hemispheres:

$$SDR = \begin{Bmatrix} x^2+y^2+z^2=1 \\ x_1x+y_1y+z_1z \geq 0 \end{Bmatrix} \cap \begin{Bmatrix} x^2+y^2+z^2=1 \\ x_2x+y_2y+z_2z \geq 0 \end{Bmatrix} \cdots \cap \begin{Bmatrix} x^2+y^2+z^2=1 \\ x_Nx+y_Ny+z_Nz \geq 0 \end{Bmatrix} \tag{3-4}$$

This can be represented as following:

$$SDR = \{p(x,y,z) | x^2+y^2+z^2=1,\ x_1x+y_1y+z_1z \geq 0, i = 1,2,\ldots,N\} \tag{3-5}$$

Siddique and Rosen [Sid 97] presented a four steps algorithm related with SDR determination. The algorithm consists in:

- Step 1: Determination of the vectors in SDR.
  For any vector ***r*** (*x*, *y*, *z*) in the whole space, normalize the vector to satisfy the equation (3-4), then check whether the unite vector meets all the inequality conditions. For example, if in a point (x_1,y_1,z_1), $x_ix+y_iy+z_iz<0$, vector ***r*** cannot be a feasible removal direction. This will just need $O(n+1)$ time calculation for checking the feasibility of a direction.
- Step 2: Determination of the bound of the removal space.
  For each pole, first normalize the pole (here *pole* stands for vector) and then find rotation to move pole away from the *Oxy* plane such that the points are still in the hemisphere that can determine the required rotations. Then use central projection to map all the points corresponding to the poles onto the z=1 plane, which can construct the largest convex containing all the poles based on the convex hull algorithm. At last, map the points that define the boundary of the convex hull back to the unit sphere.

- Step 3: Explicitly constructing the feasible removal direction space.
  First: for each plane, defined by Step 2, compute the intersection with adjacent plane which can be done by cross vector product of the normal vectors to the two planes.
  Second: the order of the intersection can be determined by traversing the planes in the same order as the convex hull, by using the Step 2.
- Step 4: Feasible removal directions for fits mating conditions.
  First: gather the axes of all the fits mating conditions.
  Second: check if all the axes are parallel. If so, determine if the axes lie in SDR. According to the mating conditions of the target component with its surrounding parts, the SDR for this target component are computed. If axes lie in SDR, the target component may be disassembled.

A method for automatic generation of the disassembly path in a virtual environment was proposed by [Mo 02]. The method's algorithm is similar to this of Siddique and Rosen [Sid 97].

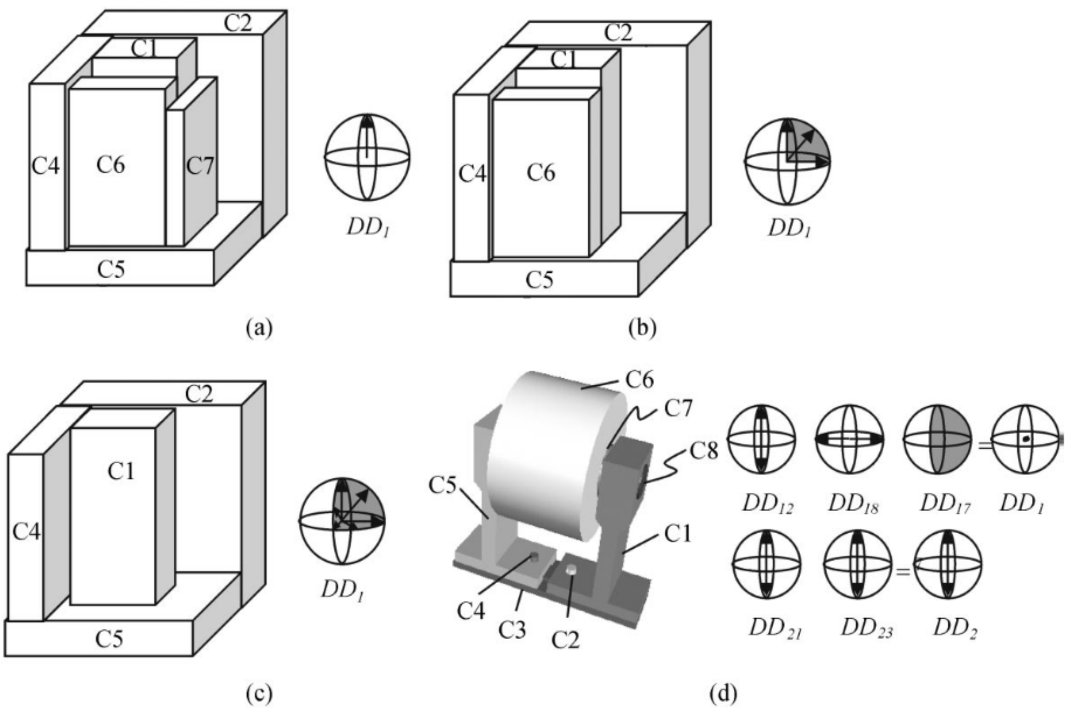


Figure 3.2 Disassembly directionality [Mo 02]

3.2.3 Approximation of the shape

In order to automatically get SDR from the CAD design of the product, its shape should be modeled as polyhedron. However, not all the 3D models are composed with only polyhedral shapes. Thus, a method for non-polyhedral shape using shape polyhedron approximation was proposed by Pomares et al. in [Pom 04]. Fig. 3.3 shows the approximation of the shape of a screw to a polyhedron taking different resolutions into consideration and using different qualities for the representation.

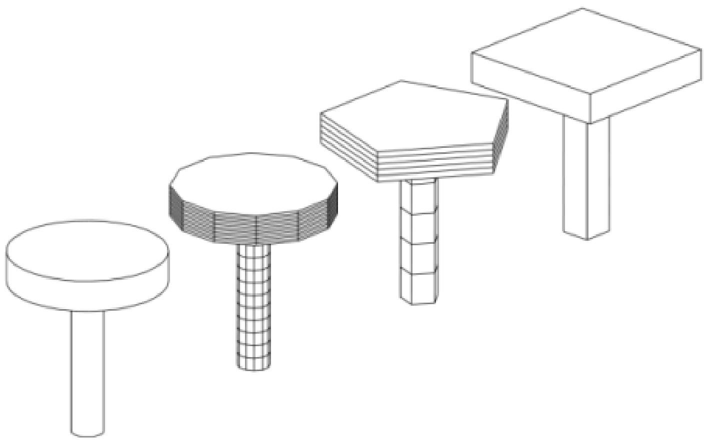


Figure 3.3 Approximation of the shape of a screw to a polyhedron [Pom 04].

Every polyhedron can be presented by a set of intersected flat surfaces. To each surface a normal vector v, directed outwards from the polyhedron is associated. Beside it, a point is required in order to get the complete equation for the polyhedron. For any surface, Si, two parameters are necessary in order to define the plane which can be presented as $(\overrightarrow{v_{i1}}, P_{i1})$, where 1 indicates the initial position of the surface of the polyhedron. Thus, for any translation matrix T, the new values for the assembly/disassembly removing direction of each surface are:

$[\overrightarrow{v_{i1}}, \quad p_{ij}]^T = T \cdot [\overrightarrow{v_{i1}}, \quad p_{ij}]$, where *T* represents the set of translation that the surface has undergone.

With our method, the constraint directions are divided into two types. The first one is the *SDR*, which stands for the possible removing direction of the part in the assembly. The other one is *collision detection* during disassembly in the range of *SDR*. These two aspects are presented in the next.

3.2.4 Geometric feasibility

The set of direction for removal (SDR) concerns the freedom of the components in the assemblies. In order to disassemble a component from an assembly, after having its SDR, the component also needs free path for its disassembling without collision with other parts. If two subassemblies of components can be assembled or disassembled without collision this situation is called Geometric feasibility. The latter represents the free path of assembly and disassembly [Su 07]. In order to explain the concept of geometric feasibility, Fig. 3.4 shows some typical constraint directions. The arrows represent the limits of the possible removal directions of part B according to part A.

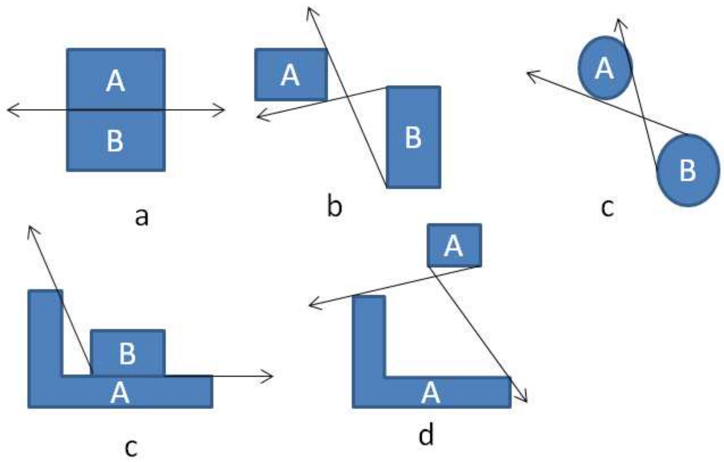


Figure 3.4 Typical constraint directions

a) Basic notions and definitions

The relationships among parts can be described using the following concept. Let consider a box composed by two parts (body B and cover A), and parts C, D and E inside thus defining the assembly N= {A,B,C,D,E} (see Fig. 3.5).

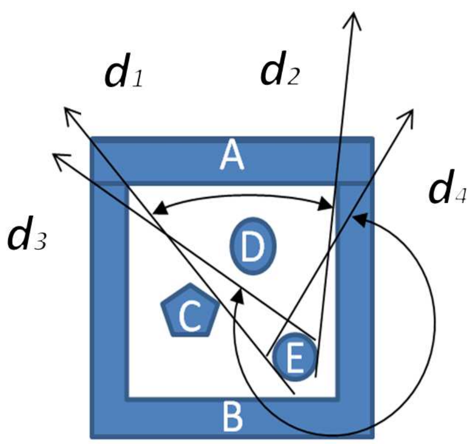


Figure 3.5 An assembly's draft

As shown in this figure the constraint directions that limit to remove E out of the box are d_1 and d_2 (limited by the Cover A) and d_3 and d_4 (limited by the body B). Thus, the relationships among parts can be described using the following definitions:

Constraint directions (*CD*) are the set of possible moving directions of a part limited by the dimensions of another part. The CD of part E according to parts A and B for assembly N= {A,B,C,D,E} can be presented respectively as:

$$CD_{EB} = (d_1, d_2)$$

$$CD_{EA} = (d_3, d_4)$$

Therefore, for an assembly, the constraint direction S= {A, B} of the assembly state to part E can be calculated as:

$$CDES = (d_3, d_4) \cup (d_1, d_2) = [0, 2\pi) = I$$

If the directions set I = [0, 2π), it implies that E is completely constrained in all the directions in S.

The constraint direction of certain assembly state S can be calculated by

$$CD_{is} = \bigcup i \epsilon s CD_{ij} \qquad (3\text{-}6)$$

Constraint assembly state (CAS)

The constraint assembly state is the state that a part is completely blocked in an assembly. It allows to measure the sets of constrain direction for a particular part labeled, $CAS_E^1 = \{A, B\}$ where 1 means the minimal number of parts (here A and B) for the constraint assembly state of the particular part E. If we consider the other parts which may bloc E (here C and D), the respective Constraint assembly states are:

$CAS_E^2 = \{A, B, C\}$ and $CAS_E^3 = \{A, B, C, D\}$

It is easy to find out that CAS_E^1 is the smallest. Thus, it is called the minimal constraint assembly state of E. It can be used for the possibility evaluation of disassembly. If the minimal constraint assembly state of a part is less than the directions set I= [0, 2π] for a component in an assembly, this component can be disassembled.

3.2.5 Collision detection

After obtaining SDR, the next issue is the detection of collisions which may occur during the disassembly process. The Extrusion operation method, proposed by Cameron in [Cam 90], is one of the most common way for collision detection. The Extrusion operation involves transforming the collision detection problem into an intersection detection problem over the 3D space. Note, that the disadvantage of this method is its high cost of explicit computation. In order to avoid this explicit computation, other methods use Projection calculation [Jim 01] instead of the extrusion operation. The projection calculation method, consists in projecting the volume of the parts onto a lower-dimensional subspace in a given projection direction. This leads to swept volume approach as shown in Fig 3.6 [Jim 01]. If the swept volumes for the objects (parts) in a scene do not intersect in a given projection direction, there is no collision happening among them in this direction. Thus, the projection is defined as the direction of the extrusion. Projection allows the transition from 3D to 2D representation. Note, that the projection process is static, meaning that there is no motion of the parts. Only projections are done in order to pass from 3D to 2D representation. Thus, if there is collision in 2D this means that there will be a real collision between the 3D parts.

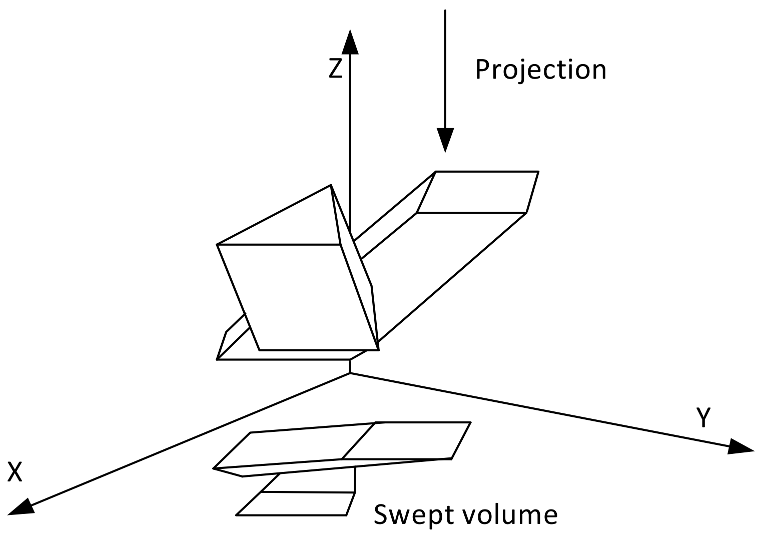


Figure 3.6 Swept volumes for two components.

As previously said, by using the projection method, a transition from 3D to 2D representation appears. Thus, for a product the lower (2D) dimensional subspace in this situation is like a convex hull, as shown in Fig.3.7. Only the components that can be moved out of the convex hull can be disassembled in a direction of SDR. If swept volume has no intersection with any other components when the projection reaches the convex hull, this component can be disassembled. This is the component N° 5 as shown in Fig.3.7. It is defined by its biggest dimension, here the distance AB, which limits the bandwidth of the projection area of the SDR.

Let us note that a component in a product cannot be disassembled for two reasons. The first one is that there is no SDR. The second reason is the presence of collision with other components in the direction of projection. Those two aspects are integrated in the method proposed in the next sections of this chapter.

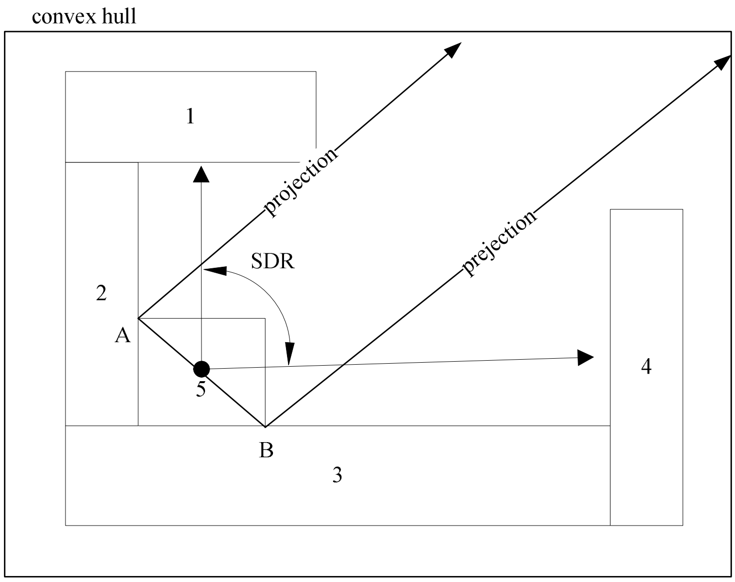


Figure 3.7 Set of directions of removal (SDR) and Collision Detection

3.3 Disassembly Geometry Contacting Graph definition

Based on modular product design, the consideration of disassembly of modular units is important. In reality, sometimes it is possible to separate several modules and reach the target component instead of disassembling the components one by one. If this is the case, each module is considered as one component. Thus, the proposed method is based on two main steps. The first one consists in building the Disassembly Geometry Contacting Graph (DGCG) of the product. The second one consists in generating of disassembly sequences.

3.3.1 Disassembly Geometry Contacting Graph (DGCG)

It is assumed that if parts are welded they appear as one complex part in the graph. Thus, building the DGCG allows minimizing the complexity of the model for disassembly sequence generation.

The DGCG aims to divide the components related with the targets into different disassembly levels according to their abilities to be disassembled. For example, if some components can be disassembled directly, without removing other components, we called them 1-st-disassembly level components. Consider a product (assembly) containing *m* components. For each of the components, the SDR and the collision detection are checked. Then, after m iterations, the 1st-disassembly level components1 are obtained. Again, recheck the remaining components on the condition of the 1st-disassembly components have been already removed. Thus, the 2-nd-disassembly components are obtained and so on.

In order to generate the sequences according to DGCG graph, the reasons why some components cannot be disassembled in the preceding levels need to be checked as well. For example, in Fig 3.8, component 4 can be disassembled in level 2. Thus, the reason why component 4 cannot be disassembled in level 1 needs to be recorded. Component 7 can be disassembled in level 3. The reasons why it cannot be disassembled in the upper levels 2 and 1 need to be recorded for the later sequences generation analysis and so on. When the target component is reached the process for building the DGCG stops automatically.

Therefore, the key points of the proposed method are that not only it obtains the disassembly level for each related component but also involves the reasons why a component cannot be disassembled in the prior levels.

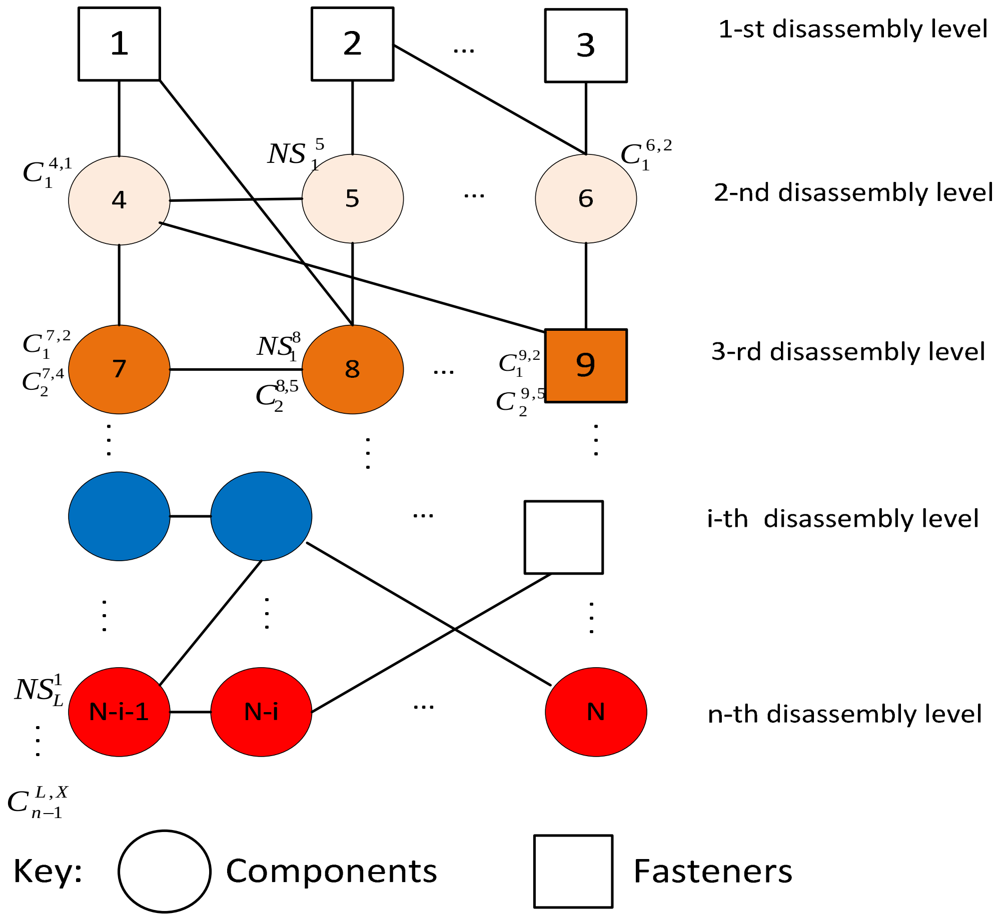


Figure 3.8 A general case of DGCG.

[1] In order to simplify the terminology the "*i-th-disassembly* level components" will be called simply "*i-th-disassembly* components"

As shown in Fig 3.8, the edges between the 1-st-disassembly components and the other remaining components are cutting off in order to obtain a new SDR. However, the 1-disassembly components need to be included in the collision detection as well. For example, if we want to get collision information about component 4, the edge between 4 and 1 is cut off. Thus, component 1 will be moved out and components 2 and 3 will still remain there for collision detection checking.

The following notations are involved in the graph:

- if component i cannot be disassembled in level n because of *collision* with the component j, it is labeled by $C_n^{i,j}$,
- if component i cannot be disassembled in level n because of *no SDR*, it is labeled by NS_n^i.

Conventionally, the components from the same disassembly level are represented by the same color in the DGCG. In order to reduce the complexity, and consequently, the computational effort and time, it is supposed that the fasteners (specified by squares in the graph) can only move in one direction. They do not *need* to be involved in the calculation of SDR because, by definition, they are supposed to have SDR. So, for a fastener in the assembly, the SDR calculation results will only be in one direction along with the centre line of the fasteners (screw, bolt), as shown in Fig 3.9.

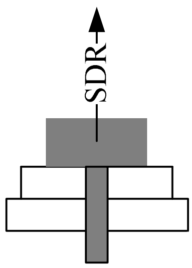
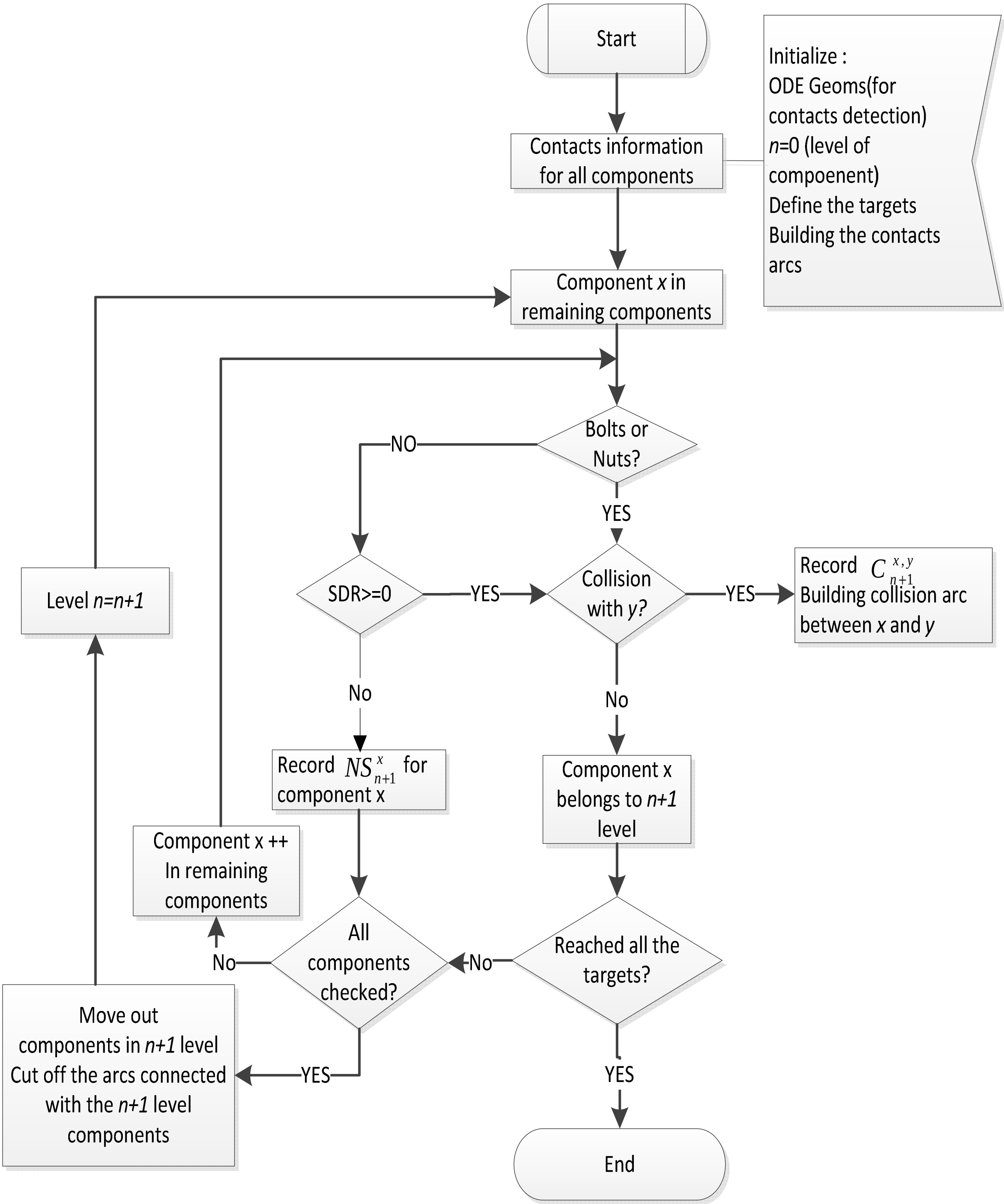


Figure 3.9 Set of direction for removal (SDR) of fasteners.

Generally the fasteners should be the 1-st-disassembly components. If it is not the case, it means that they have collided with some other components. As seen from Fig.3.8, component 4 cannot be disassembled in the first level because of the collision with component 1, labeled as ($C_1^{4,1}$). Component 8 cannot be disassembled in level 1 because of no SDR, labeled as (NS_1^8). It cannot be disassembled in level 2 either because of the collision with component 5 labeled as ($C_2^{8,5}$). The detailed flow chart for generating the disassembly geometry contacting graph (DGCG) is shown in Fig. 3.10. It consists in three main steps:



legend: ODE Geoms (Open Dynamics Engine library)

x, *y*, … stands for any component in the product

Figure 3.10 Flow chart for DGCG generating

- First, the 3D component models of the assembly are imported in the realized software through a XML file coming from a CAD software. Each model is followed by ODE (Open

Dynamics Engine) Geoms model which is used to detect the contacts among the components (see Chapter 4). Then the contacts' arcs among the components are built.

- Secondly, the analysis of the components' type and the collisions are performed. If the component is not a fastener, check the SDR. If the component is a fastener, just the collision information is checked. If it has a collision with some parts, build the related collision arcs and record that the component cannot be disassembled in this level because of the collision. If there is no collision, the component can be disassembled in this level.

- Third, removed components can be disassembled in the upper levels, cut off the arcs, recheck the remaining components again and so on.

Note that the process for building the DGCG, stops when all the targets appear in it. Based on the flow chart presented in Fig. 3.10 a pseudo code associated to the method was developed for implementation (Table3.1)

Table 3.1 Pseudo code of disassembly geometry contacting graph building

Initialization

n= 0: level

x, y…: any component

A : any product for disassembly

∀ Target x∈A , following "Ode Geom", then Building contact lines according to the contactin information.

Loop: x ==bolts or nuts?

Loop1: If yes, collision with any other component (y)?

If yes, record $c_{n+1}^{x,y}$

Else, component x belongs to n+1 level

Loop2: All targets reached?

If yes, End

Else the entire remaining components reached?

If not, x++, (go to loop)

Else, cutting components (n+1 level), x points to one of the remainin components,

(go to loop)

Else, SDR>=0?

If yes, go to collision detection (go to loop1)

Else, record NS_{n+1}^{x}*, go to the entire remaining components reached (go to loop2).*

3.3.2 Three Micro-units

The next step consists in generating the disassembly sequences according to DGCG. For this purpose three cases, called micro units, which consider all the possible situations of relationships, among the components in the DGCG are addressed (Fig. 3.11). Suppose x is the target component.

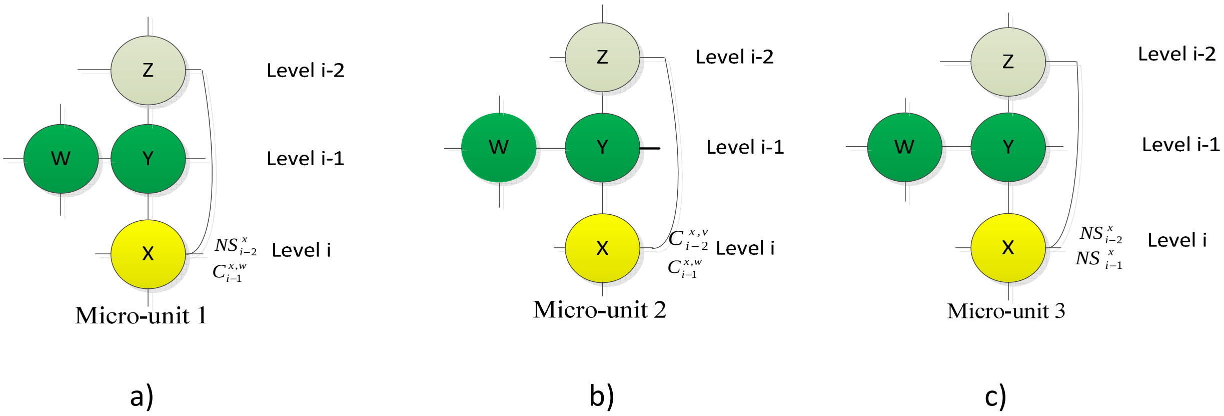
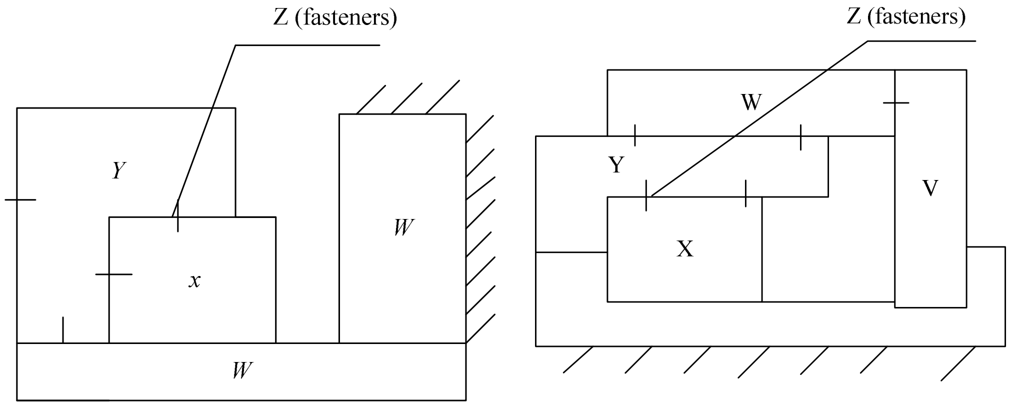


Figure 3.11 Three types of Micro units for DGCG building.

Case*1*: Micro-unit *1*. Transition from No SDR (*NS*) to Collision (*C*).

In micro-unit 1 (Fig. 3.11a), suppose the target component x is in collision with component w in level (i-1), labeled as $C_{i-1}^{x,w}$. Suppose also it has no SDR, in level (i-2), labeled asNS_{i-2}^{x}. Component w has to be moved before the target component x because of $C_{i-1}^{x,w}$. Therefore the next target, called auxiliary target, should be component w. However, if the component w is in the lower level instead (means that component w cannot be disassembled before component x), in this case, component y should be the auxiliary target because it connects with component x. Although x has a collision with w, but after removing component y, component x can change its direction of disassembly. Therefore, the component y removing cancels the collision between x and w. As seen from Fig 3.11, component x has changed its status from NS_{i-2}^{x} in level (i-2) to $C_{i-1}^{x,w}$ in level (i-1). It means that component x cannot be disassembled in level (i-2) because of no SDR. After removing the components in level (i-2), component x cannot be disassembled because of collision in (i-1) level ($C_{i-1}^{x,w}$). Therefore, the components in level (i-2) connected with the target component x are responsible for this change (called also transition). Thus, component z should be disassembled first to reach component x.

In order to illustrate this case, a relatively simple example is shown in Fig. 3.12a. The target component *x* is connected with component *y* by two fasteners *z*. Component *w* is the ground component defined as the components upon which, stands the rest of the assembly. In order to obtain SDR, fasteners *z* should be moved first. Then component *x* will have a collision with component *w*. However, it is not possible to remove *w* before component *x* because component *w* is the ground part. Therefore component *y*, being in contact with *x*, should be disassembled instead. After removing component *y*, component *x* will have new SDR thus avoiding collision with component *w*. Consequently, the disassembly sequence is x→{z,y,x}. In this notation, *x* is the target component, {z,y,x} is the feasible disassembly sequence.



a) b)

Figure 3.12 Examples for micro-units.

Case 2: Micro-unit 2. Transition from Collision (*C*) to Collision (*C*)

In micro-unit 2 (Fig. 3.12b.), suppose target x is in collision with w in level (i-1) labeled as $C_{i-1}^{x,w}$. Suppose it has also collision with component v, in level (i-2), labeled as $C_{i-2}^{x,v}$ in . (Rq. Component v, which stands for any component in the product, is not shown in the figure). If component v is supposed to be component w, the auxiliary target should be w. If component v is distinct from w and its disassembly level is the same or upper than the level w (it means the component v can be disassembled before or at the same time with the component w), v should be the next disassembled component. For example, in Fig. 3.12b, target x will have collision with component v after removing fasteners z between x and y. In the same way, according to the relationships between y and x, after removing y, component x will have collision with w. After removing component y, the collision of x will change into collision with w. As the removal of component y causes this transition, it should be disassembled before w. If the disassembly level of component v is lower than the level w, component v will be ignored. The components connected with the target x in level (i-2) should be responsible for this transition from $C_{i-1}^{x,w}$ to $C_{i-2}^{x,v}$. Both y and w should be the targets in order to get x. Component y should be the auxiliary target before x. Thus, the disassembly sequence is: x→{z, y, w, x}.

Case 3: Micro-unit 3. Transition from no SDR (NS) to no SDR (NS)

If the target component x (in level *i*) cannot be disassembled in level i-1 because of no SDR labeled NS_{i-1}^{x}, all the components connected with x in the upper levels should be the next auxiliary targets. Because x cannot get SDR, it is necessary to move its surrounding components (to get the SDR). As shown in micro-unit 3 (Fig. 3.11c), components y and z should be the next auxiliary targets. This means that if the target does not have collision with any components, the components in the upper levels connected with the target should be disassembled in order to get the SDR.

The pseudo codes for the three types of Micro units are shown in Table 3.2.

Let us note that there is no case of transition from Collision (C) to no-SDR (NS). In fact, when collision happens, it means that the moving component already had its SDR in the considered level, and consequently it is not possible for it to change into No SDR in the lower levels. Let us note also that, for any target component connected with fasteners, the latter should be disassembled first, if it is possible of course. If fasteners cannot be the 1-st-disassembly components, it means that they have collision with some other components. Consequently, these collision components should be disassembled before the fasteners.

Table 3.2 Three micro-unit pseudo codes

Micro-unit 1	*Micro-unit 2*	*Micro-unit 3*
A: stands for any product for disassembly *y: stands for component connected with component x in i-2 level*		
For ∀Target x ∈A , in i level *take* $C_{i-1}^{x,w}$, *If level (w)<level(x),* *return target component v* *else: return target compon* *y* *end for*	*For ∀Target x ∈A , in i level* *Take,*$C_{i-1}^{x,w}$ *and* $C_{i-2}^{x,v}$, *If w==v, return w,* *else if level(v)<=level(w), return v* *else return targets components* *connected with x in (i-2) level* *end for*	*For ∀Target x ∈A , in i level* *For j=0 to i:* *Return target components* *connected* *with x in j level* *J++* *end for* *end for*

Legend: level (x): the level of part x

According to the proposed three cases the disassembly sequences can be performed. For instance, if component 8 is the target component (see Fig. 3.8) it cannot be disassembled because of no SDR in level 1 (NS_1^8). It also has collision with component 5 at level 2 ($C_2^{8,5}$). This situation belongs to case 1. As component 1 is fastener connected with 8, it should be moved first to get SDR. Then component 5 must be moved according to case 1. But, in order to get component 5, component 2 must be moved first according to case 3. Thus, the sequences will be {1,2,5,8} or {2,1,5,8}. As components 1 and 2 belong to the same (here first) level, both sequences (1,2) or (2,1) are possible. Consequently, the sequences order for target component 8 is 8→{(1,2),5,8}. In this notation, the components in brackets mean that they can be disassembled in any order, here (1,2) or (2,1).

Based on the three cases, addressed here above, the flow chart for determination of the feasible disassembly sequences is shown in Fig.3.13.

The proposed method gives the feasible disassembly sequences, according to the least level of disassembly, which may result in many sequences if there are two or more components lying in the same level. For example, if two components are connected by fasteners, any of them may be disassembled firstly if they have not collision with some other components of course. For this situation, it is difficult even meaningless to decide the bolts disassembling order.

Compared with other methods, our method is more efficient. It may be evaluated by its complexity $O(sm(n\text{-}1))$ which is lower than the computation complexity $O(sm^2)$ of the wave propagation method as mentioned in §3.1. Our method removes the useless sequences as the problem for product disassembly is transferred from an invert tree search problem. For a complex product, for instance, if the target component is in the upper level of disassembly, the lower level components can be deleted directly thus reducing the computational effort and time for sequences generation. Based on the Flow Chart for Disassembly sequences generation presented in Fig. 3.13 a pseudo code associated to the method was developed for implementation (see Table 3.3).

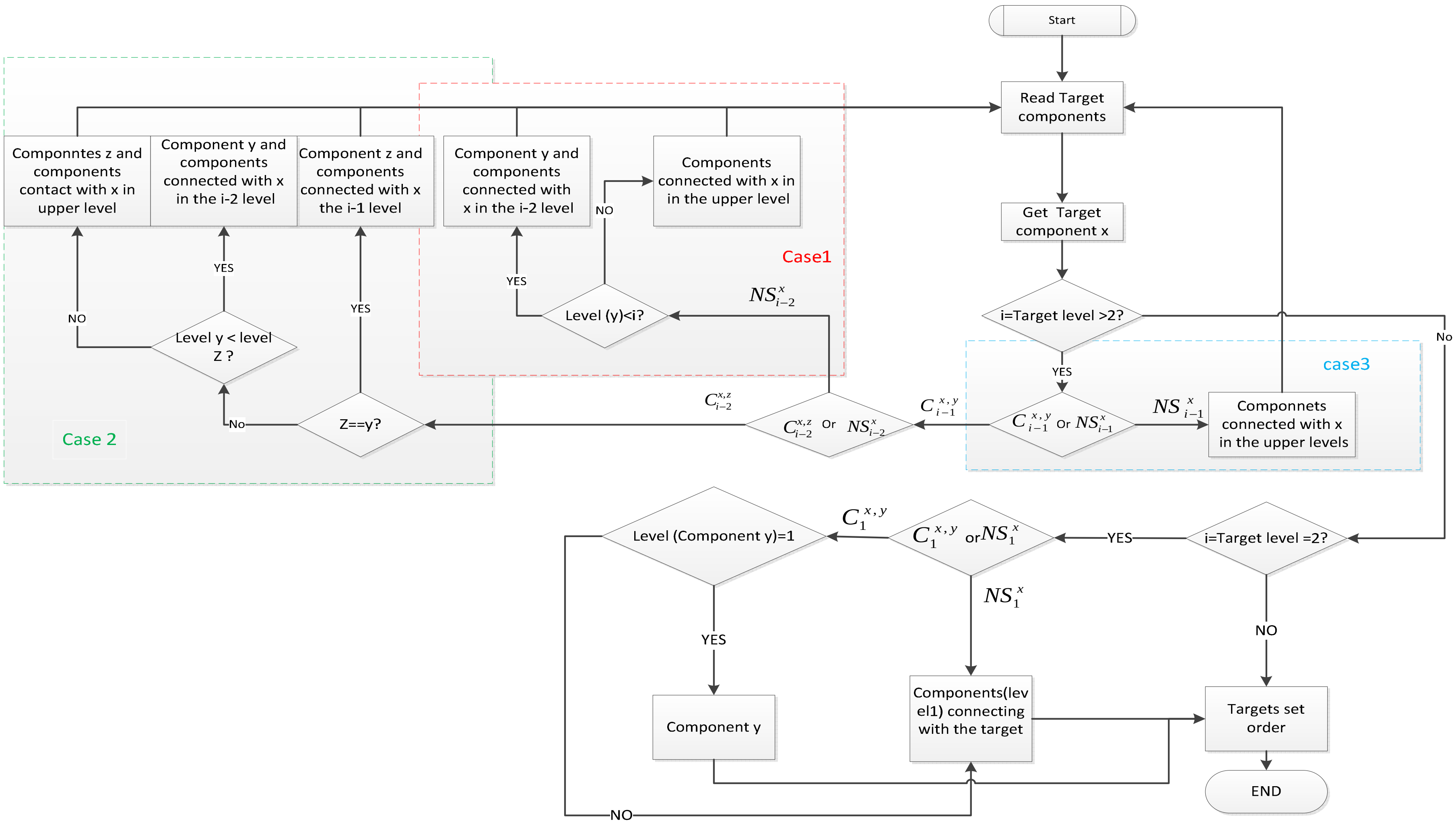


Figure 3.13 Flow Chart for Disassembly sequences generation

Table 3.3 Pseudo code for sequences generation.

B: stands for the targets list
x,v : any component
For ∀Target x ∈B ,
i= level (x) > 2?:
If yes, $C_{i-1}^{x,y}$ *or* NS_{i-2}^{x}*?*
If $C_{i-1}^{x,y}$*:* $C_{i-2}^{x,z}$ *or* NS_{i-2}^{x}*?*
If $C_{i-2}^{x,z}$*: Micro-unit 2 (Table 2)*
Else NS_{i-2}^{x}*: Micro-unit 1(Table 2)*
Else NS_{i-2}^{x}*: Micro-unit 3 (Table 2)*
Else, i==2?:
If yes :$C_{1}^{x,y}$ *or* NS_{1}^{x}
If $C_{1}^{x,y}$*: level(component y)==1?*
If yes: component y is first (target components order)
Else: components in level 1 connected with x (target components order),
Else: NS_{1}^{x}*: components in level 1 connected with x (target components order),end*
Else: target components order
End for

3.4 Cases studies

The proposed method was tested for disassembly operations simulation of mechanical and electromechanical assemblies with different degrees of complexity. Two examples: electrical motor (Fig. 3.14) and the wrist of a five degrees of freedom robot arm (Fig. 3.15) are presented here below.

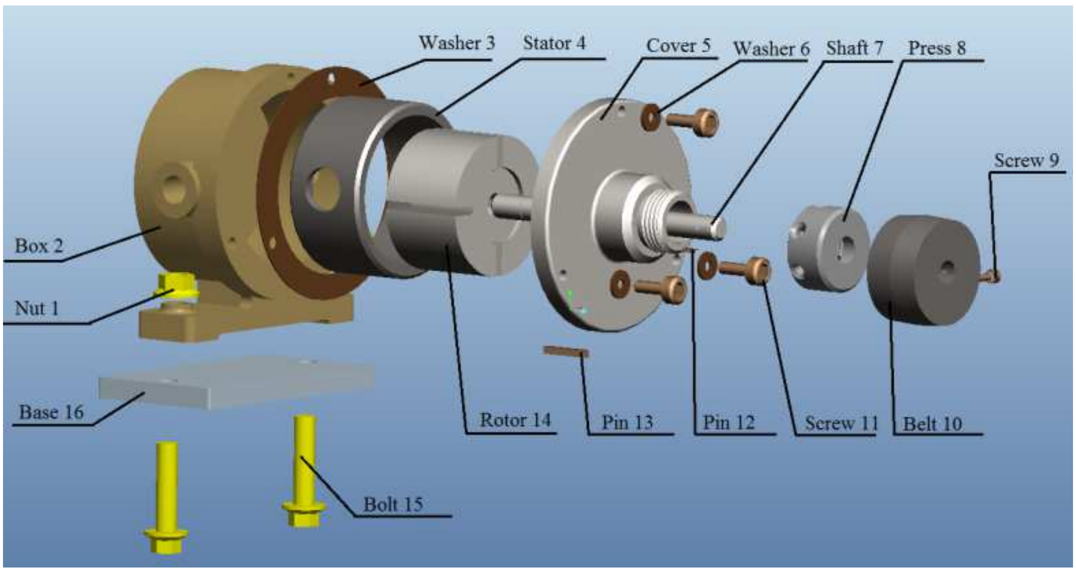


Figure 3.14 An example of electrical motor with sixteen components

As previously said, the fasteners connecting two parts are counted as one connection. Thus, in Fig.3.14 screw 11, for example, stands for all the screws connecting Cover 5 and Box 2. In Fig. 3.15 bolt 3, for example, stands for all the bolts connecting coupling 4 and cover 2.

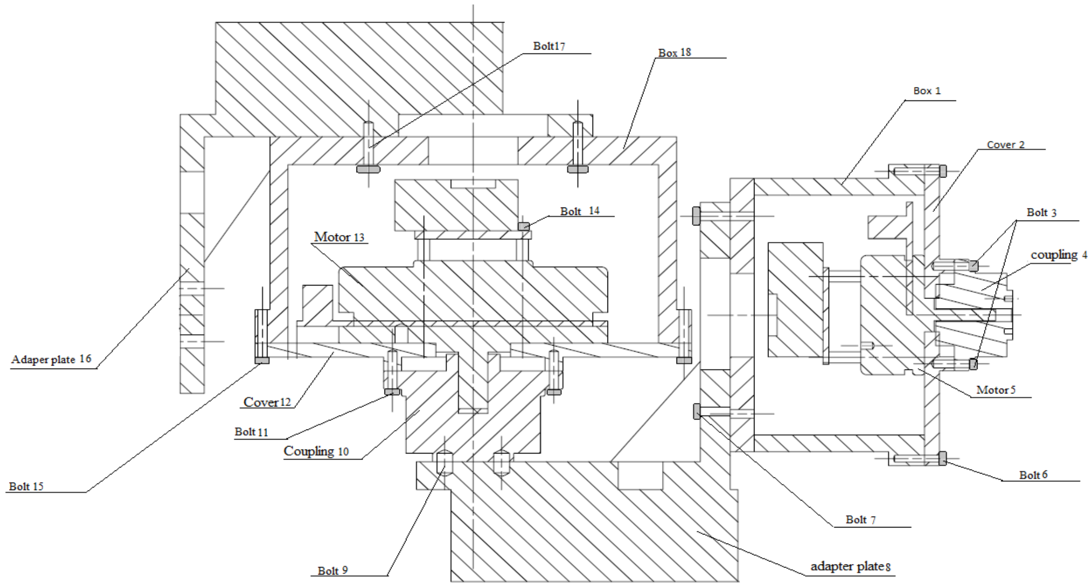


Figure 3.15 Five degree of freedom robot arm with eighteen components

If the target components are respectively cover 5 for the electrical motor (Fig. 3.14) and the Motors 5 and 13 for the robot arm, thanks to the proposed method the disassembly process is performed by the following two steps:

3.4.1 Building the Disassembly Geometry Contacting Graph (DGCG)

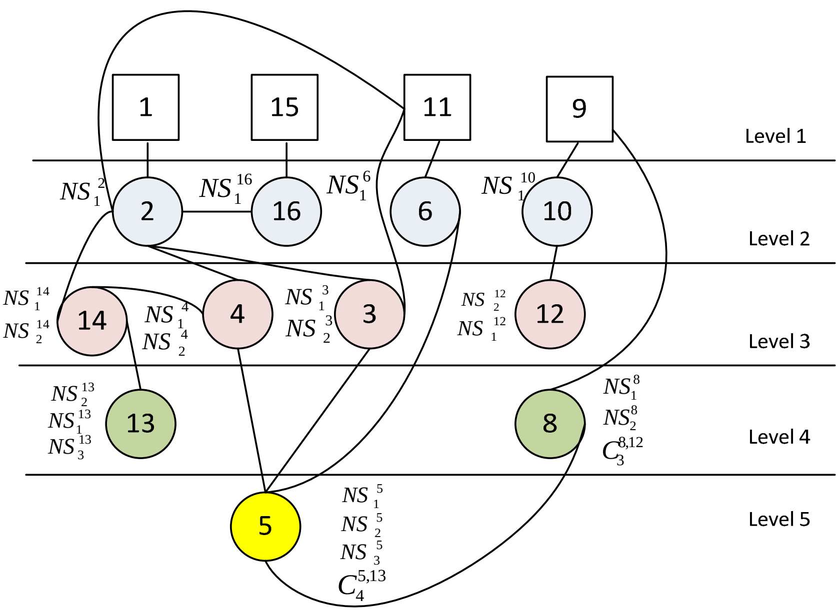


Figure 3.16 DGCG for Cover 5 of the electrical Motor.

According to the relations among the components in the assembly, and the flow chart for disassembly geometry contacting graph (DGCG) generating (Fig. 3.10), the realized computer application allows building the associated DGCGs. Thus, the two five levels DGCG for the electrical motor and the robot arm are built as shown in Fig. 3.16 and Fig 3.17 respectively.

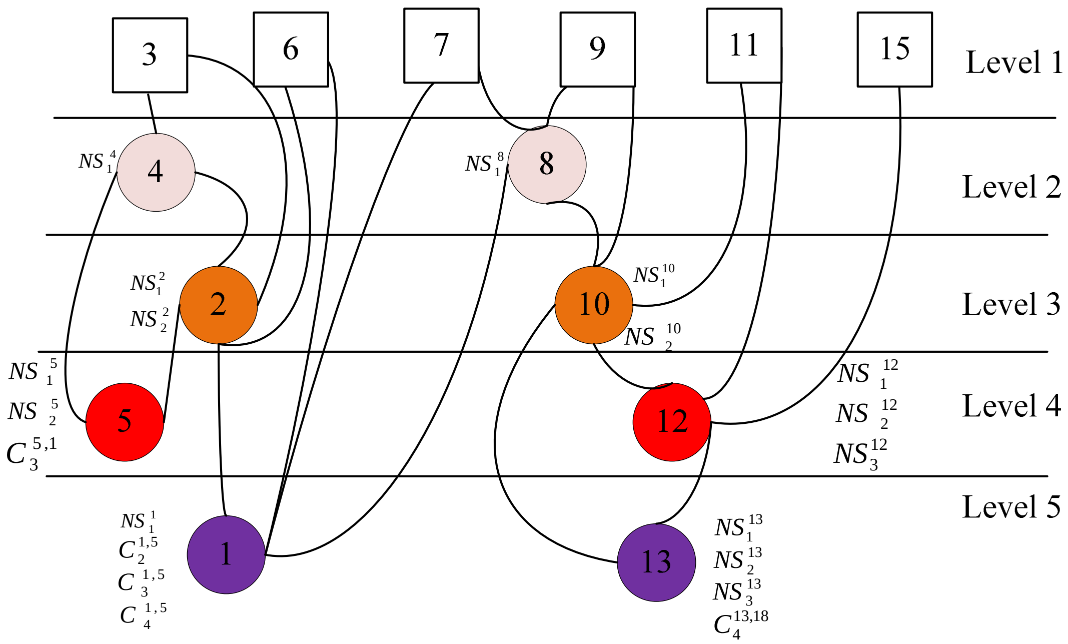


Figure 3.17 DGCG for Motors 5 and 13 of the five degrees of freedom robot arm.

3.4.2 Sequences generation for one target of electrical motor

As previously said, the associate DGCG (Fig. 3.16) is assimilated like an inverted tree. From the graph, it is seen that the target component 5 can be disassembled in level 5. It cannot be moved in level 4 because of the collision with component 13 ($C_4^{5,13}$). Therefore, according the case 1, the next target component should be component 13.

In levels 3, 2 and 1, the component 5 does not have SDR, therefore, according the case 3, components 3, 4 and 6 (connected with the target) should be disassembled first.

Then, component 13 is the target. From the DGCG in Fig.3.16 is seen that component 13 cannot be disassembled in the upper level because on No SDR. Therefore, the next target should be 14 (connected with 13). The reason that target 14can not be disassembled in the upper levels is the same as the component 13, namely No SDR. Therefore, the next targets should be components 2 and 1.

For component 3 disassembling, according to the case 3, components 2 and 11, connected with 3 should be removed previously. For component 6 disassembling, component 11 should be disassembled first. All these relationships amongst the parts in the DGCG are presented as disassembly order graph (DOG) in Fig. 3.18. DOG is generated manually. It allows generating the disassembly order for the target component in the inverse arrow side.

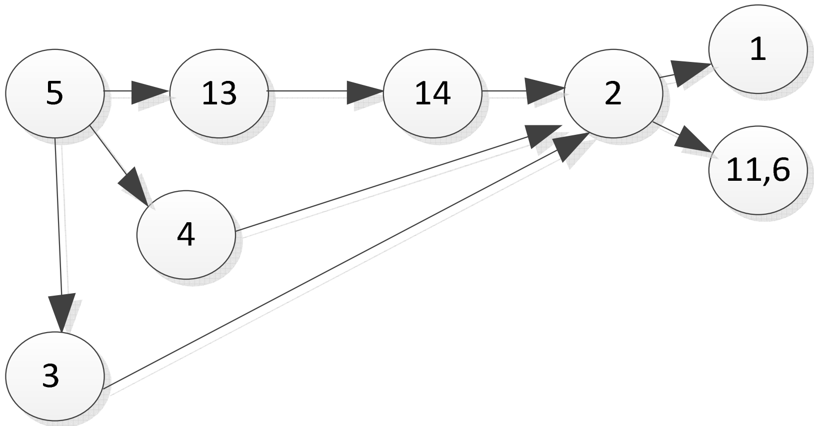


Figure 3.18 Disassembly order graph for component Cover 5 (see Fig. 3.14).

The twenty four possible disassembly sequences generated by the realized Python computer program, based on the proposed DGCG method are presented in Fig. 3.19. According to the relationships amongst the components, all these sequences are possible. However at this stage of the study we cannot evaluate the best one.

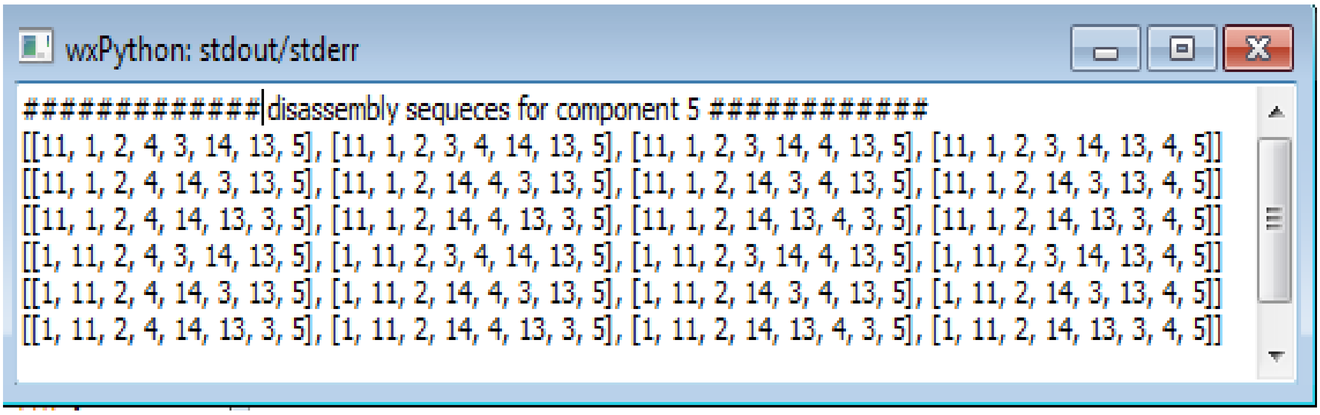


Figure 3.19 Possible Disassembly Sequences for Cover 5.

3.4.3 Sequences generation for two targets of robot arm

Let us start by the first target component, namely Motor 5 in Fig. 3.15. The reason that it cannot be moved in level 3 is its collision with component 1 ($C_3^{5,1}$). In the upper levels (number of level smaller than 4), there is no component 1. Therefore, according to case 1, the next target component should be component 2, which is connected with 5. In this case, removing the contact between 2 and 5 provides the other direction for disassembling component 5 allowing to stop the collision between 5 and 1, as described in case 1. Therefore component 5 cannot be disassembled in level 3 because of component 2 which cancels the collision between component 5 and component 1 ($C_3^{5,1}$). Component 5 cannot be disassembled in level 2 either because of the No

SDR (NS_2^5). The transition from (NS_2^5) to ($C_3^{5,1}$) happens after removing components 4 and 8. Note that only component 4 has contact with the target component 5. Therefore component 4 is responsible for this change and its removal provides the SDR for component 5. Consequently, the disassembly sequence for component 5 is 5→ {2,4,5}.

Then, component 2 is the target, called auxiliary target, because its level is lower than the level of component 4. Thus, component 2 can be disassembled in level 2. According to the DGCG (see Fig. 3.17), it has No SDR in the 1-st and 2-nd levels (NS_1^2, NS_2^2), which belongs to the Micro-unit 3 described in case 3. Therefore, all components connected with component 2 in the upper levels should be the auxiliary targets. Bolts 3 and 6 in level 1 are connected with component 2. Therefore they need to be removed firstly. After removing these bolts, component 4 moving provides SDR for component 2. Thus, the sequence should be 2→{(3,6),4,2}.(3,6)}.

Then component 4 becomes the auxiliary target, which can be disassembled in level 2 as seen in Fig. 3.17. Note that it cannot be disassembled in level 1 because of the No SDR (NS_1^4). According to case 3, component 3 removal can provide the SDR. Therefore, the sequence is 4→{3,4}.

According to the above description, the DOG for target component Motor 5 is built as shown in Fig. 3.20(a). Note that it can be simplified in the so called reduced graph by drawing out, in the lower level, components 3 and 4 as shown in Fig. 3.20(b). Thus, the disassembly sequence for target 5 is 5→{(3,6),4,2,5}.

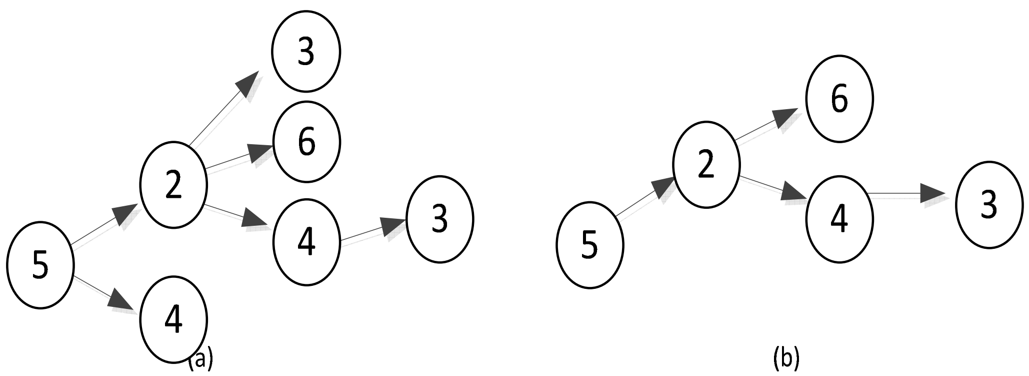


Figure 3.20 Disassembly order graph for component 5(a) and its reduced graph (b).

The same analysis can be done for the other target component, namely Motor 13, which has: collision with component 18 in level 4 ($C_4^{13,18}$) and No SDR in level 3(NS_3^{13}) as shown in Fig. 3.17. The component 18 is not shown in the DGCG, because the calculation algorithm for DGCG building stops when the targets, here components 5, and 13, are reached. Which means that component's 18 disassembly level is much lower than the targets level. According to case 1, component 10 removal gives the SDR for component 13. Component 12 removal provides free of collision movement for component 13. Consequently, the sequence should be 13→{12,10,13}.

Concerning the auxiliary target 12, it cannot be disassembled because of No SDR in levels 1, 2, and 3 ($NS_1^{12}, NS_2^{12}, NS_3^{12}$) . According to case 3 and case 4, the sequence should be 12→{10,(11,15),12}.

Concerning the auxiliary target 10, it cannot be disassembled in levels 1 and 2 because of No SDR (NS_1^{10}, NS_2^{10}). According to case 3 the sequence should be 10→ {8, (9,11),10}.

Finally, for the auxiliary target 8, it cannot be disassembled in level 1 because of No SDR (NS_1^{8}). Thus, the sequence should be 8→ {(7,9),8}.

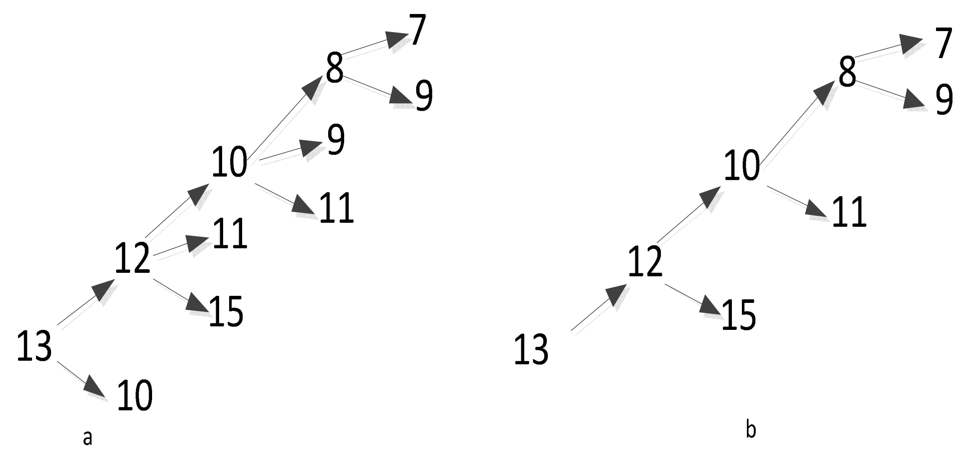
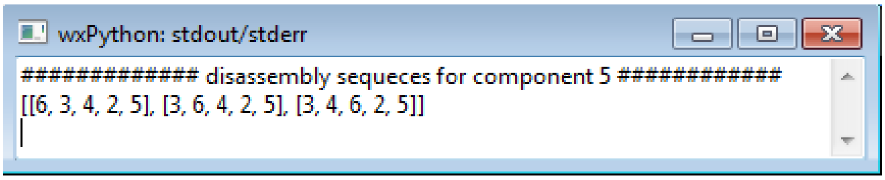


Figure 3.21 . Disassembly order graph for component 13 a). and its associate reduced graph b).

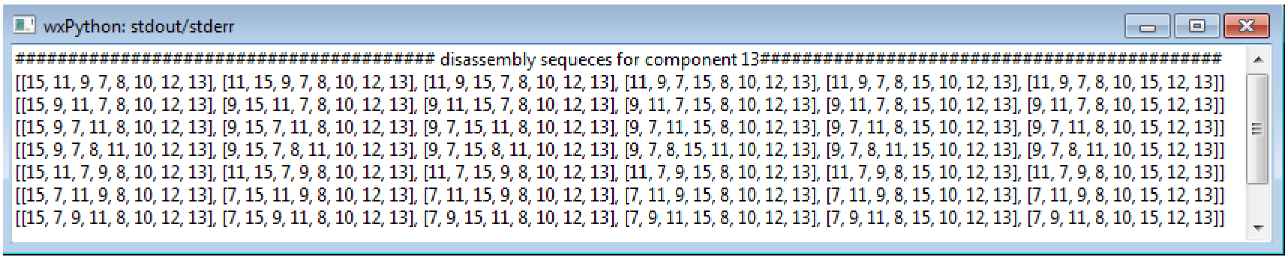
Consequently, for target 13, the disassembly order graph and its associate reduced graph are shown in Fig.3.21 (a) and Fig.3.21 (b) respectively. Thus, the sequences for disassembly of component 13 is (see Fig. 3.21 b): 13→{(7,9),8,11,10,15,12,13}.

The input 3D assembly models are based on VTK (Visualization Toolkit) library and acquired through a VRML files coming from CAD software. The contact identification is based on ODE Geom (Open Dynamics Engine) libraries (see Chapter4). The results of generating the feasible sequences for target components 5 and 13 for the five degrees of freedom robot arm are shown in Fig. 3.22. Note, that there are three possible sequences for target component 5 (Fig. 3.22a) and forty eight for target component 13 (Fig. 3.22b).

According to the proposed method only the related components with the target are considered, and the process stops automatically when the target component is reached. For target 5 for instance, only components 2, 4, 3 and 6 appear in the sequences, all of its unrelated components are removed from the graph thus minimizing the model complexity and search time. It may be noted, that the computation resource and consequently *cpu* time are related with the number of components (m) in an assembly, and the number of targets (s) to be disassembled.



(a) disassembly sequences for target Motor *5*



(b) disassembly sequences for target Motor *13*

Figure 3.22 Possible disassembly sequences for Motors 5 and 13.

For example, the wave propagation algorithm of Srinivasan and Gadh [Sri, 99a] is of complexity O(sm2). However, our method's complexity is O(sm(n-1)),where n is the number of level in the graph allowing reaching the target. The disassembly level n is far less than the number of components m (n<m) thus allowing reducing computational effort and time as we claimed in the Introduction of this Chapter (Section 3.1).

3.4.4 Summary

The disassembly order graph is like a problem of inverted tree containing a minimum set of components related with the target component disassembly. Thus, the unrelated components are eliminated in order to reduce calculation resources. For the example of *Cover 5* (Fig. 3.14) there are *15* components involved in the DGCG. If components *10* or *6*, for instance, are supposed to be the targets, there will be only *8* components involved in the DGCG. Thus, the twenty four possible sequences are generated according to the *DGCG*, based on the least level of disassembly method.

The reason for these different possible sequences generated is the presence of more components in the same disassembly level that can be removed in any order. However, in the real disassembly process, the purpose of the upper level components disassembly is to get the lower level component, which means if there is a chance to disassemble the lower level component, it should be disassembled first. Thus, for *Cover 5* it took 30ms *CPU* for its sequences generation.

3.5 Conclusion

In this Chapter a new method for disassembly sequences generation, we called "least level of disassembly graph method" is presented. Sequences' generation is based on the notion of disassembly geometry contacting graph DGCG. The graph is built on the collision and SDR detection analyses for each given component in an assembly. With the investigated three cases, the method eliminates all the components unrelated with the targets.

The DGCG model contains a minimum set of components related with the target. Thus, the unrelated components are eliminated in order to reduce the computational resource. Our method can generate the sequences for any kinds of complexity of products. With DGCG, the possible sequences are easily to be generated considering the least level of disassembly.

As we previously said, if some of the components are grouping in modular units (modules) every module can be considered as one component thus simplifying the DGCG graph and consequently reducing sequencing search time.

Chapter 4

Virtual Reality Environment for disassembly simulation: sequences generation and evaluation

This chapter presents the basic concept of the virtual reality environment upon which the application for disassembly operations simulation (generation and evaluation) is realized. First, the 3D graphics pipeline in general is presented. Then, the key technologies and devices of the developed virtual reality disassembly environment (VRDE) based on Python programming language and utilizing mixed VTK (Visualization Toolkit) and ODE (Open Dynamics Engine) libraries are detailed.

4.1 Introduction

Virtual reality (VR) technology plays a vital role in simulating advanced 3D human-computer interaction by providing users with different kinds of sensations (visual, auditory, haptic, …). Virtual disassembly simulations allow designer to evaluate concepts in virtual environments during the early stages of product design. With virtual prototyping applications, the optimal design process for *design for assembly* (DFA) can be incorporated easily in the conceptual design stage. Using haptic or auditory technology, allows designers to interact with the parts with the human basic emotions. Thus, collision detection and contact force are transmitted to the operator in real time.

4.2 3D graphics pipeline in general

In order to present the 3D graphic systems the basic and some more important concepts related with 3D graphics are addressed here below. Various transformations in 3D graphics consist in taking an object in 3D and displacing it on the screen while keeping the illusion of depth in the scene. The common way to perform these transformations for each step of the pipeline is to use vectors or matrix calculation, which are presented here below.

4.2.1 Right-Handed and Left-Handed coordinate systems

In VTK library (see Section 4.3) all the rotations and translations are depending on the utilized coordinate systems. For this purpose the later are recalled briefly.

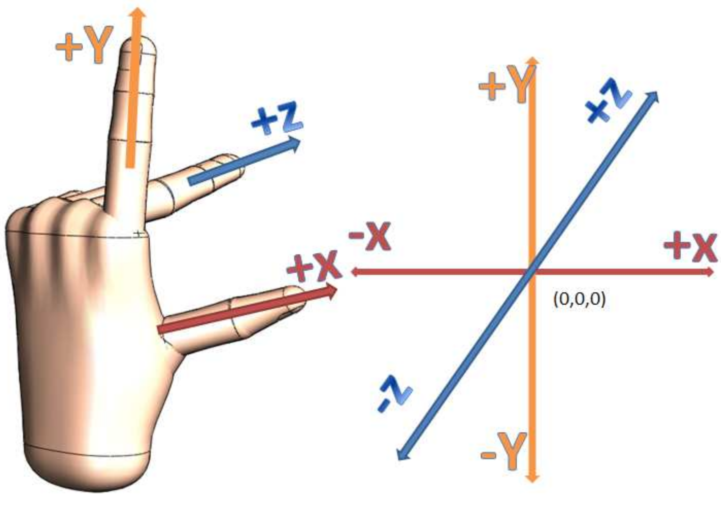


Figure 4.1 Left-Handed coordinate systems.

For 3D objects visualization most of the existing 3D graphic Systems use the classical "*Cartesian Coordinate System*" which main property is that the cardinal axes are perpendicular. It consists in.

- Stretch left Arm and form a 90° angle with Elbow.
- Point with Thumb to the right side (+*x*).
- Point with Pointing Finger up (+*y*).
- Point with Middle Finger in *z* direction.

If the middle finger is pointed to the +*z* direction, the hand is forming a left-handed coordinate system as shown as Fig. 4.1. On the contrary, if the direction is the –*z* direction, it forms the right hand coordinate system. The coordinate system is very important concept for 3D graphics, because all the matrix calculations related with transformations (rotations and translations) are based on the coordinate system. For example, positive rotation is clockwise about the axis of rotation in the left hand coordinate systems. Positive rotation is counter-clockwise about the axis of rotation in the right hand coordinate systems.

Most of typical 3D graphic libraries for example OPEGL and VTK based are using right hand coordinate system. Normally Direct 3D library uses the left hand coordinate system instead.

4.2.2 Coordinate systems in 3D scene

- Graphical overview of the Transformation

The process of displaying a 3D scene in computer graphics is assimilated like taking a photo with a camera. There are four matrix transformations among four different coordinate spaces as shown in Fig.4.2. The transformation process consists in:

1. Put the objects (or models, or avatar) in the world (*Model Transformation* or *World transformation*).
2. Define the Position and orientation of the camera (*View transformation*).
3. Select the camera lens (wide angle, normal or telescopic), adjust the focus length and zoom factor to set the camera's field of view (*Projection transformation*).
4. Showing the image on a selected area of the object (*Viewport transformation*)

In computer graphics, the transform for a vertex *V* from one coordinate space to another space *V'* is carried out by multiplying the vector with a *transformation matrix M*, i.e., $V' = M\ V$.

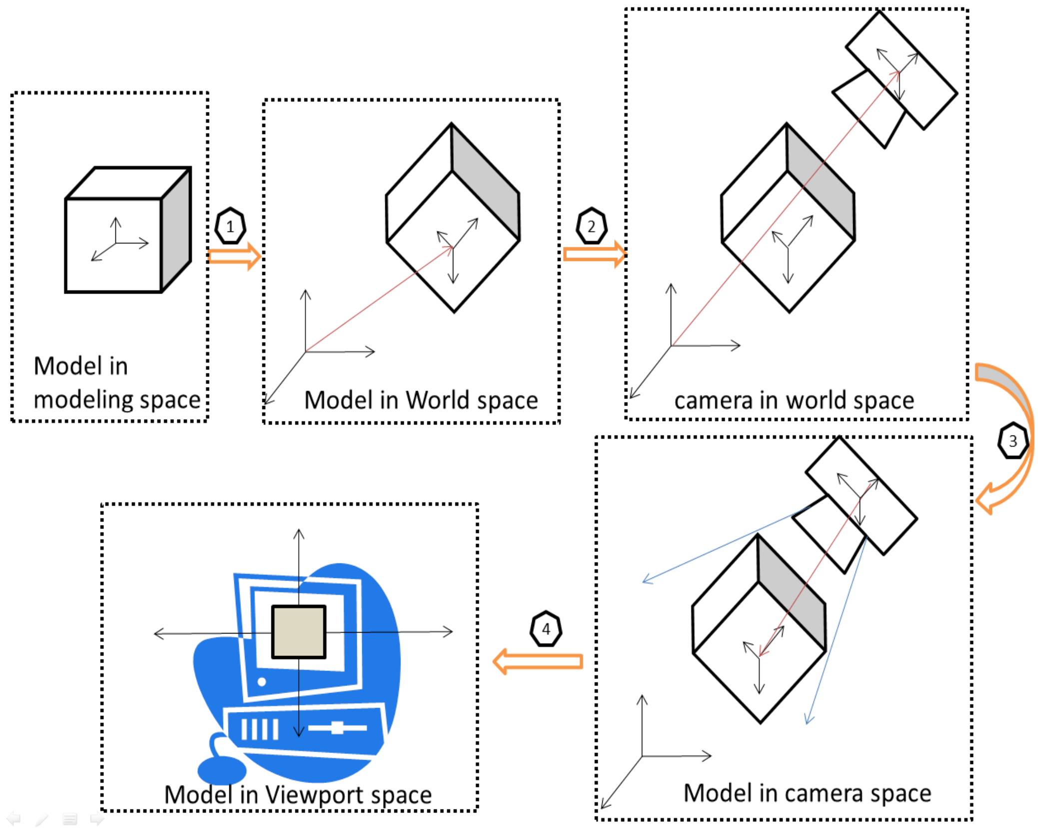


Figure 4.2 Transformation pipeline overview

4.2.3 Model or World transformation

Each object in a 3D scene is defined by its own coordinate system, named as its *model space*. Model (or World) transformations allow to place an object anywhere within the 3D world. They can change the position (translation), orientation (rotation) or size (scaling) of an object as shown in Fig. 4.2. This figure shows a cube that has to be first rotated about its center, and then translated to the position in the world frame. This is known as the *model transformation* of *world transformation*. The latter consists in scaling, rotation and translation of an object in order to match the dimension of the world. Note that the transformation is presented by the basic movements of translation, rotation and scaling.

- **Translation**

Let us consider the example for the transformation of a vertex *v*. In *(x, y, z)*, coordinate system, the vertex can be presented as a vector:

$$v = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4-1}$$

This classical presentation is based on the right hand coordinate. Note, that for left hand coordinate, it can be presented as row vector. In the next, all the calculations, if not specified, are based on right hand coordinate system.

Let us suppose to move vertex *v* to a distance $L = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix}$. Its new position is:

$$\mathrm{L}v = \begin{bmatrix} x + l_1 \\ y + l_2 \\ z + l_2 \end{bmatrix} \qquad (4\text{-}2)$$

For matrix calculation there is no properly matrix that can be used directly. Therefore, homogenous coordinates representing of a 4-coordinates vector is used.

The vertex $v' = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$ has an additional forth *w*-component of 1. If w≠1, then (*x*, *y*, *z*, *w*) corresponds to Cartesian coordinates (*x*/*w*, *y*/*w*, *z*/*w*). If *w*=0, it represents a vector, instead of a point (or vertex). If the vertices are represented in the 4-component *homogeneous coordinates* (*x*, *y*, *z*, 1) the *homogeneous* matrix is:

$$T = \begin{bmatrix} 1 & 0 & 0 & l_1 \\ 0 & 1 & 0 & l_2 \\ 0 & 0 & 1 & l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4\text{-}2)$$

Therefore,

$$T \cdot v' = \begin{bmatrix} 1 & 0 & 0 & l_1 \\ 0 & 1 & 0 & l_2 \\ 0 & 0 & 1 & l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} l_1 + x \\ l_2 + y \\ l_3 + z \\ 1 \end{bmatrix} \qquad (4\text{-}3)$$

where the last column of matrix T: $L = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \\ 1 \end{bmatrix}$ is the translation vector.

- **Rotation**

  For the unit circle (Fig.4.3):

  $x_1=\cos(a_1)$, $y_1=\sin(a_1)$, $x_2=\cos(a_1+a_2)$, $y_2=\sin(a_1+a_2)$.

  $x_2=\cos(a_1+a_2)=\cos a_1\cos a_2 - \sin a_1\sin a_2=x_1\cos a_2-y_1\sin a_2$

  $y_2=\sin(a_1+a_2) = \sin a_1\cos a_2 + \cos a_1\sin a_2 = x_1\sin a_2+y_1\cos a_2$

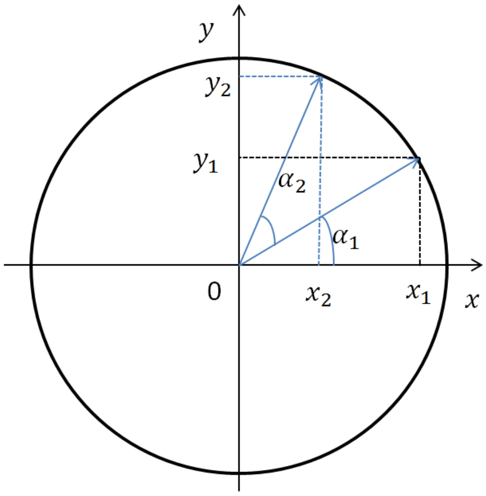


Figure 4.3 Rotation in the unit radius cycle

The rotation matrix around *z* axis is:

$$R_z = \begin{bmatrix} \cos a_2 & -\sin a_2 & 0 & 0 \\ \sin a_2 & \cos a_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{So, } R_z \cdot v' = \begin{bmatrix} \cos a_2 & -\sin a_2 & 0 & 0 \\ \sin a_2 & \cos a_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} xcosa2 - ysina2 \\ y\, cosa2 + x\, sina2 \\ z \\ 1 \end{bmatrix}$$

For 3D rotations about *y* and *x* axes the rotation matrices are respectively:

$$R_y = \begin{bmatrix} \cos a_2 & 0 & -\sin a_2 & 0 \\ 0 & 1 & 0 & 0 \\ \sin a_2 & 0 & \cos a_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos a_2 & -\sin a_2 & 0 \\ 0 & \sin a_2 & \cos a_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Scaling**

The purpose of scaling transformation is to either increase or decrease the size of the object. A 3D scaling can be represented in a 3x3 matrix:

$$S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix}$$

where s_x, s_y and s_z represent the scaling factors in x, y and z directions, respectively. If all the factors are the same, it is the so called *uniform scaling*.

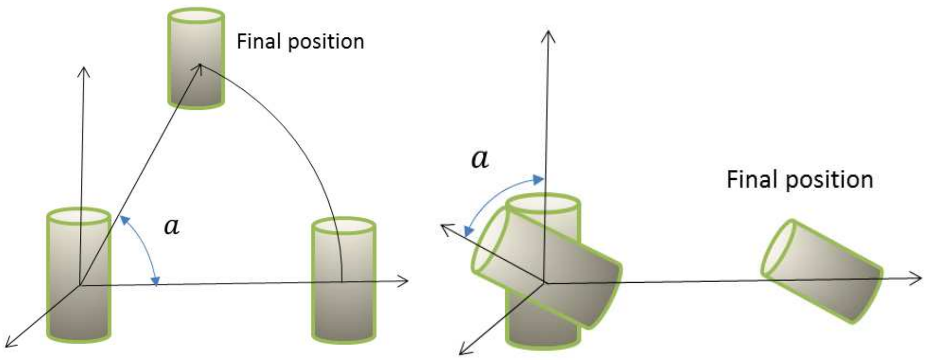
The transformed result $\boldsymbol{V'}$ of vertex $\boldsymbol{V}$ can be obtained via matrix multiplication, as follows:

$$\text{S}.v= \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x.s_x \\ y.s_y \\ z.s_z \end{bmatrix}$$

- **Combination of the transformations**

In most cases it is necessary to scale the object in order to fit it with the 3D world: rotate it into the required orientation, move it somewhere, etc. In order to perform the above series of transformations the vertex position have to by multiplied by the first transformation matrix and then the obtained result to be multiplied by the next transformation and so on.

Thus, a successive affine transformations (R_1, R_2, T_1, T_2, T_3 ...) operating on a vertex V can be computed via concatenated matrix multiplications $V' = ...T_3T_2T_1\mathrm{R}_1\mathrm{R}_2\mathrm{R}_1\ V$. Note, that the order of matrices is influencing the results of the position of 3D object. In 3D graphics it is common to scale the object first, then to rotate it and following by a translation then apply camera transformation and finally project it to 2D.



(a) Translation first. (b) Rotation first.

Figure 4.4 Rotations and translations.

Let us consider the translation first. In this case, it is difficult to set the object position in the world. In fact, when moving the object away from the origin and then rotate it, it goes around the origin which actually means that we translate it again as shown in Figure 4.4(a).

By rotating first and then translating we disconnect the dependency between the two operations as shown in Fig.4.4 (b). This is why it is always better to model around the origin as symmetrically as possible. That way when later we scale or rotate there is no side effect and the rotated or scaled object remains symmetrical as before.

4.2.4 View transformation

After arranging the objects in the 3D world, the next task is to define the camera position in the World space. This process is called *view transformation* (see Fig.4.2 ②).

In the most application cases, there are two ways for moving one object in the virtual environment. One is moving the object itself as presented in Section 4.2.5 here above. The other way is moving the position of the camera. In reality, we want to have freedom to place the camera anywhere in the world and project the vertices in a 2D plane in front of it. This will reflect the correct relation between the camera and the object on the screen. So, if it is necessary to move the camera, there are two steps to do it.

The first one is to translate the camera to the original position of world space which is easy to realize. If the camera position is (d_1, d_2, d_3) and the translation transformation is ($-d_1$,$-d_2$,$-d_3$), the associate homogenous matrix of the view is:

$$T_{view} = \begin{bmatrix} 1 & 0 & 0 & -d_1 \\ 0 & 1 & 0 & -d_2 \\ 0 & 0 & 1 & -d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In this way, the camera is in the original position of world space.

The next step consists in rotating the camera toward the target in world space coordinates. In fact, it is necessary to find out the location of the vertices in the new coordinate system that the camera defines. Therefore, $x_2 = \overrightarrow{OC} \cdot \overrightarrow{0x_c}$.

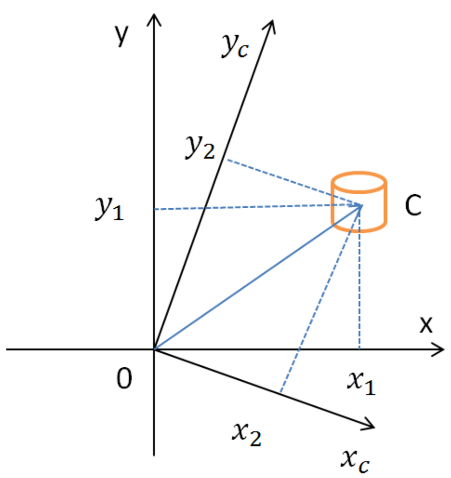


Figure 4.5 Coordinate transformations

In 3D graphics, the camera is positioned onto the world space by specifying three *vectors* in world space. For this solution, called *UVN camera* the position of the camera is defined by the following vectors (Fig. 4.6):

- *N*(*Nx, Ny, Nz)* - The vector from the camera position to the target. This vector corresponds to the Z axis (labeled by N "Normal").
- *V*(*Vx,Vy,Vz)* - The upside vector from operator's head to the sky if the camera is standing upright. This vector corresponds to the Y axis (labeled by V "Vertical").
- *U*(*Ux, Uy, Uz*) - The vector points from the camera to its "right" side" when the camera is pointed at the target, in such a way that N,V,U (here U corresponds to the X axis) form Direct-Ortho-Normal coordinate system.

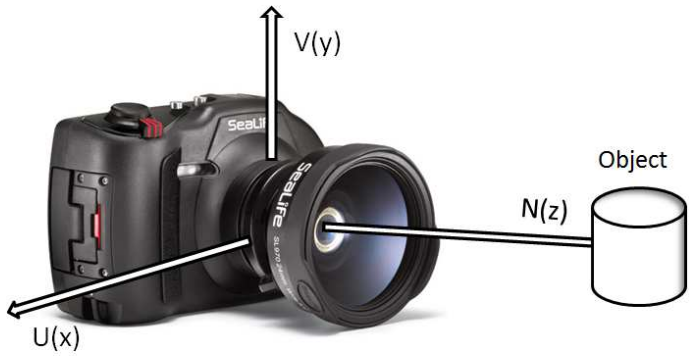


Figure 4.6 Camera space

The view homogenous matrix (in rotation only) is:

$$R_{view}=\begin{bmatrix} Ux & Uy & Uz & 0 \\ Vx & Vy & Vz & 0 \\ Nx & Ny & Nz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally, the view homogenous matrix combining the two operations (rotation and translation) is:

$$M_{view} = R_{view}T_{view}=\begin{bmatrix} Ux & Uy & Uz & 0 \\ Vx & Vy & Vz & 0 \\ Nx & Ny & Nz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & -d_1 \\ 0 & 1 & 0 & -d_2 \\ 0 & 0 & 1 & -d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4.2.5 Projection

After moving the camera, the next issue is to define what can be seen from the scene. This is done by selecting a projection mode (perspective or orthogonal) and specifying a viewing volume or *clipping volume*. Objects outside this volume are clipped out of the scene and consequently cannot be seen.

Before introducing the clipping volume, there are two notions that need to be defined namely: *projection plane* and *projection window*. The projection plane is a plane which is parallel to the Oxy plane in the camera space. Obviously, not the entire projection plane is visible. Only stuff in a rectangular area (called projection window) can be seen which has the same proportions of the screen.

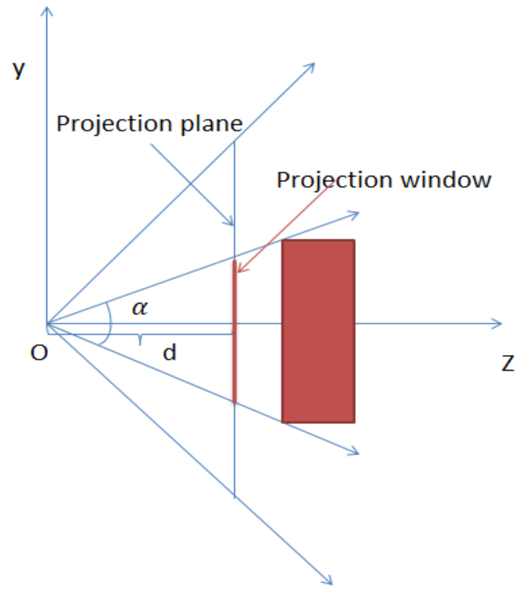


Figure 4.7 Projection plan and window

In general, the height of the screen *w* is defined into the unit size. Therefore, the height of the window will be 2. The distance between the projection planes and the camera is *d.* It is obviously that $\tan\left(\frac{\alpha}{2}\right) = 1/d$ =>$d = 1/\tan(\frac{\alpha}{2})$. For a given point (x, y, z) in the 3D world we want to find its projected coordinates (x_w, y_w) on the projection plane. For the y_w component, $y/z = y_w/d$ => $y_w = yd/z$, therefore:

$$y_w = \frac{y}{\tan\left(\frac{\alpha}{2}\right).z}$$

For the x_w component,

$x/z = x_w/d$ => $x_w = xd/z$, therefore:

$$x_w = \frac{x}{\tan\left(\frac{\alpha}{2}\right).z}$$

The size of the projection window has to be considered as well. Usually, the height of component (y_w) is normalized, thus the projected Y component is ranging between (-1, +1). Concerning the component (x_w), its width will be 2**scale*. If the height of the window is (-1, +1), the width will range between (-*scale*, *scale*). For a common 1024x728 screen for example, the *scale* is 1.333. In this way, the division by the aspect ratio has the effect of condensing the points on the x_w axis. Thus:

$$y_w = \frac{y}{\tan\left(\frac{\alpha}{2}\right).z} \tag{4-4}$$

and

$$x_w = \frac{x}{\text{scale}\cdot\tan\left(\frac{\alpha}{2}\right).z} \tag{4-5}$$

In this way, the projection window position of the objects can be gotten from their position in 3D world by eq. (4-4) and (4-5).

Note, that, z component should not influent the position of x_w and y_w. It should be used for the depth test in 3D model displaying process. The trick is to normalize the value z for all the vertices. Thus, all the positions in 3D world are divided by the z value. However, the original z value must be saved in order to perform the depth test later on. So the trick is to copy the original z value into the w value.

In this way, the components' position (x,y,z,w) in the projection window are:

$$\begin{bmatrix} x_w \\ y_w \\ 0 \\ w \end{bmatrix} = \begin{bmatrix} \frac{1}{scale\cdot tan\left(\frac{\alpha}{2}\right)} & 0 & 0 & 0 \\ 0 & \frac{1}{tan\left(\frac{\alpha}{2}\right)} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \tag{4-6}$$

It is pointed out that, the components' position in the projection window is not related with the z value. At the same time, the z value is saved into w value for later depth testing. However, the whole process is not finished yet. The remaining problem is how to use the value z in order to perform the depth testing. According to eq. (4-6), all the z will become 0. For depth testing, as shown in Fig.4.8, the z value should be in the view volume *Z-near* ≤*Z* ≤ *Z-far*.

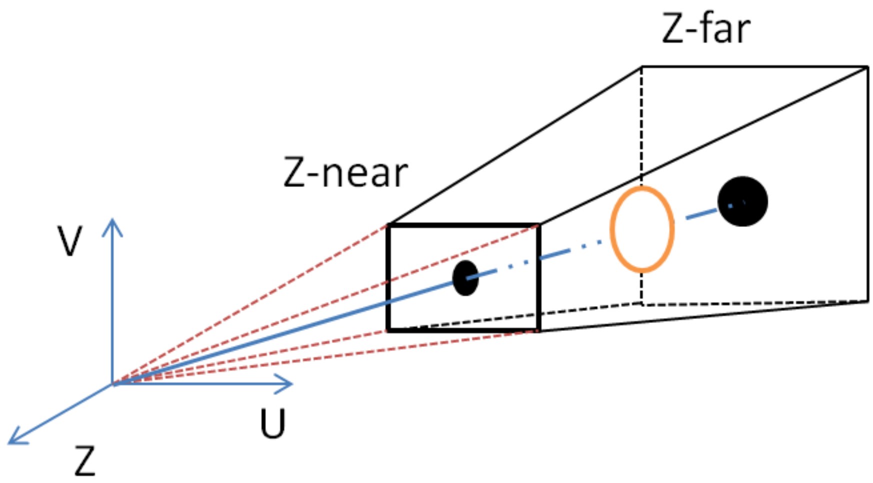


Figure 4.8 Depth testing.

After dividing all the positions of 3D model in 3D world by *z*, all the value should be mapped to *[-1, 1*] range. Let consider the function of projection *f(z)=Az +B*, where *A* and *B* are arbitrary constants to be calculated such as *A+B/*s should be in *[-1, 1]*. Thus, the last problem is to find the right A and B. As seen from Fig.4.8, when *z= Znear*, A+B/z=-1.

Therefore,

$$A+B/zNear=-1 \tag{4-7}$$

Similarly,

$$A+B/Z\text{-}far=1 \tag{4-8}$$

From equations (4-7) and (4-8) we have: B=$\frac{2\cdot zFar\cdot zNear}{zNear-zFar}$ and A=$\frac{-zNear-zFar}{zNear-zFar}$.

Consequently: Xw=p.X, or $\begin{bmatrix} x_w \\ y_w \\ ztest \\ 1 \end{bmatrix} = p\cdot\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

where: p=$\begin{bmatrix} \frac{1}{scale\cdot tan\left(\frac{\alpha}{2}\right)} & 0 & 0 & 0 \\ 0 & \frac{1}{tan\left(\frac{\alpha}{2}\right)} & 0 & 0 \\ 0 & 0 & \frac{2\cdot zFar\cdot zNear}{zNear-zFar} & \frac{-zNear-zFar}{zNear-zFar} \\ 0 & 0 & 1 & 0 \end{bmatrix}$ is the projection matrix.

4.2.6 Viewport Transformation

Firstly, all 3D objects have to be imported into world space. Secondly, they have to be transformed into the camera space. Then all the positions of 3D model have to be projected into 2D computer screen. At last, all the positions have to be shown in a rectangular display area on the screen window. This area is called *viewport* which is measured in the screen's coordinates. This viewport is defining the size and the shape of the displaying area for mapping the projected scene captured by the camera onto the window. This shape area can be or not the entire screen.

In 3D graphics, a viewport is 3D view to support *z*-ordering, which is needed for situations such as ordering of overlapping windows. The Viewport Transformation is calculated by the so called *viewport matrix*. This matrix is calculated by the product of the following three matrixes. The first one, called *reflection of y –axis*, is defined as:

$$M_1=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}$$

Then, after reflecting of all the data, it has to be scaled according to the size of computer screen or the size we want to define. The *scaling matrix* is:

$$M_2=\begin{bmatrix}w/2 & 0 & 0 & 0\\ 0 & h/2 & 0 & 0\\ 0 & 0 & maxZ-minZ & 0\\ 0 & 0 & 0 & 0\end{bmatrix}$$

The third matrix represents the translation of the data origin to the center of the screen or the place we defined before. The translation homogenous matrix is:

$$M_3=\begin{bmatrix}1 & 0 & 0 & minX+w/2\\ 0 & 1 & 0 & minY+h/2\\ 0 & 0 & 1 & minZ\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Thus, the viewport matrix is $M_{viewport}= M_3M_2M_1$.

4.3 Visualization Toolkit

After presenting the general pipeline of the 3D graphic here above, the purpose of this section is to do an overview of the Visualization Toolkit (VTK) upon which the proposed application (in C++, Python) is built.

In the application, VTK library is used for displaying 3D model. It is an open-source, freely available software system for 3D computer graphics, image processing and visualization. The central structure of the Visualization Toolkit may be represented as a pipeline of data, starting from a source of information and arriving to an image rendered on the screen.

4.3.1 Pipeline for VTK

VTK is freely available open-source system for 3D computer graphics, image processing and visualization. The object oriented VTK is rapidly becoming the standard for scientific visualization toolkits [Sch 96, Kok 07]. This is an open source class library containing a large number of functionalities for scientific data presentation.

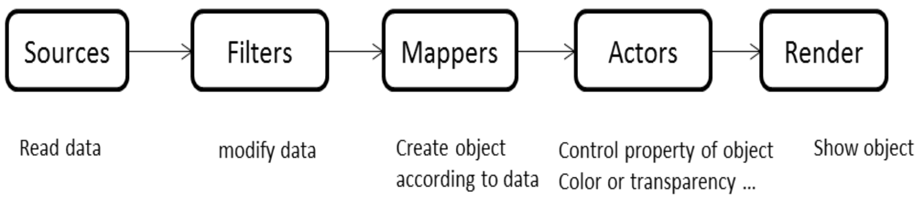


Figure 4.9 Pipeline for VTK library.

The pipeline for VTK can be described as following (Fig 4.9):

- Sources

Sources are the data needed to be shown on the screen. Basically, two kinds of sources are available for application. First are the Readers, which are used for reading data out of files in a range of formats. The other kind of sources is generated by functions or other data flow based on the input parameters (e.g. a cone source, which generates information describing a cone by its radius and height). In general, any VTK component that does not receive a flow of data from some other VTK component can be considered as a source.

- Filters

Before showing the 3D date (sources) on the screen, Filters may be used in order to modify the data in some way. For example, Filters may extract some portion of a large data set or subsample data sets from a coarser resolution to a finer resolution, and merge multiple data into a combined output. The key concept of Filters is that they can be optional components of the VTK pipeline. Thus, VTK can include more than one filter, often three or more.

- Mappers

After filtering, all data can be transferred to "Mappers". “Map” the data from source file to a physical manifestation can be performed by the rendering engine. However, sometimes it is possible to confuse Mappers with the Filters. An easy way to distinguish them is to divide the pipeline into two segments. First is the *data processing segment* including sources and filters. The second segment is the *image processing segment* which includes actors, renderers and windows. Mappers serve as the transition between the two segments. The data through the mappers is used as input for the Actors. Note, that Filters cannot be used as way for changing the sources data. The data through the filters can be used as input for the other Filters or other Mappers.

- Actors

All the data from the Mappers is used as input for the Actors. We can consider that Actors are a physical representation of the data which control the adjustment and appearance properties of the physical manifestation of the data, for example color, transparency etc.

- Render

Render and windows are the last item of VTK pipeline, which are in charge of visualization on the screen.

Note that the data function for VTK is very powerful, allowing to display any kinds of shade function as shown in Figure 4.10.

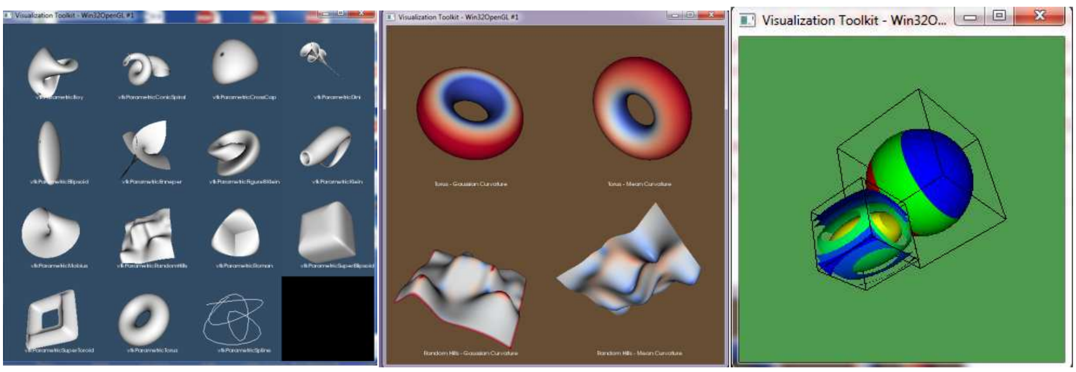


Figure 4.10 VTK examples

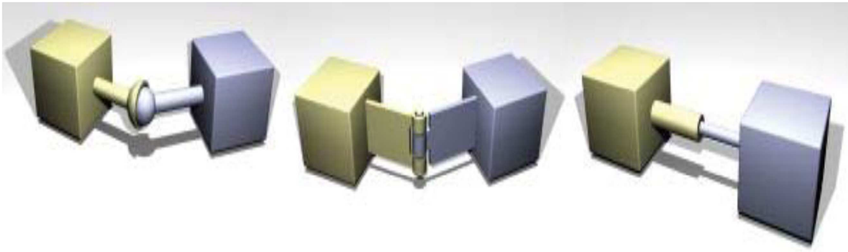
For interaction with the data, VTK employs the concepts of *picking* and *3D widgets*. Picking is used to select objects in the visualization, while widgets to interact with objects in specific ways. A widget has a visual representation within the 3D visualization. It defines the behavior that is executed when the widget is manipulated. Simple examples of 3D widgets are: the point widget for probing object information; the box widget for positioning, rotating, and scaling of objects, the spline widget for defining a spline by editing control points, etc. These entire characters of VTK are a good choice for the 3D model visualization in the virtual reality system we have developed (see Section 4.4.5).

4.4 Collision Detection Based On VTK and ODE

Fast and robust 3D collision detection algorithms are always required in the applications of Computer Graphics. As we are aware, there are four groups of algorithms for collision detection namely: *space-time intersection*, *swept volume interference*, *multiple interference detection* and *trajectory parameterization*. All of them are intended to be of practical use. The simplest decisional collision detection problem usually is described as follows: *A set of objects move over a certain time span, to determine whether any pair will come into contact.* The more intricate version always needs to find the time and features involved in the collision. These aspects are presented in the following section.

4.4.1 Open Dynamics Engine (ODE)

ODE is a free, industrial quality library for simulating rigid multi body dynamics, which is the invisible model for the collision detection and force feedback. In order to solve the problem of a polyhedral approximation, the constraint based modeling is proposed by ODE. ODE developed by Russell Smith, http://www.q12.org/ode/, is particularly good for simulating moving objects in changeable virtual reality environments. This is because it is fast, robust and stable, and the user has complete freedom to change the structure of the system even while the simulation is running. Those are the principal reasons for choosing ODE. In addition it has hard contacts, which means that a special no-penetration constraint is used whenever two bodies collide. In ODE, the joint is a relationship that is enforced between two bodies so that they can only have certain positions and orientations relative to each other. This relationship is called a constraint. Note that words *joint* and *constraint* are often used interchangeably. Figure 4.11 shows six different constraint types.



ODE “robot” joints: A ball joint, a hinge and a slider.

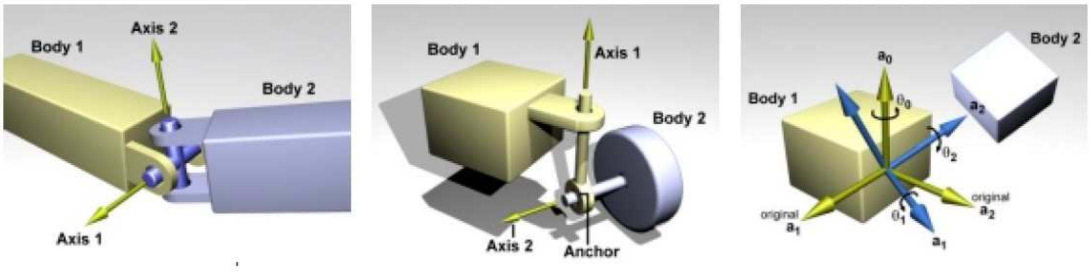


Figure 4.11 ODE’s special purpose joints. Different constraint types.

4.4.2 VTK actors connection with ODE models

In VTK, the 3D models are presented by Actors. During the disassembly process it is necessary to get the position and the orientation of the objects in ODE, and to resend them to the center of Actors in VTK in real time. In order to apply constraint forces to an object in the ODE world, a model called *Body* is created inside containing full information for the part such as: material, mass, dimensions, inertia, gravity center etc. At the same time, another model called *Geometry* is defined for presenting the shape information of the part. It is used to detect collisions between bodies and affect forces among them. In this way, the collision is detected in real time and the force feedback forces the moving path to change its directions. However, ODE does not have its own 3D objects rendering library. ODE has his own library for drawing the feature in the screen, called DrawStuff:

(http://robotics.naist.jp/~akihikoy/doxy/ode0.9/group__drawstuff.html#_details).

Note, that DrawStuff cannot meet our requirements since it is not convenient for interaction with the objects. Therefore, we provide the VTK library for the interaction with the 3D models for the interaction parts. As previous said, in VTK, the 3D models are presented by Actors. What we need to do is to get the position and rotation of objects in ODE and resend them to the center of Actors in VTK in real time. The relationships of the objects in VTK and ODE are shown as in Fig. 4.12, where two followings are appearing. The center of GEOM is following with the position of the BODY. The center of Actor is following by the Position of the ODE object. In fact the purpose is to realize the interaction of ODE objects. The world object is a container for rigid bodies and joints. Objects in different worlds cannot interact, for example rigid bodies from two different worlds cannot collide.

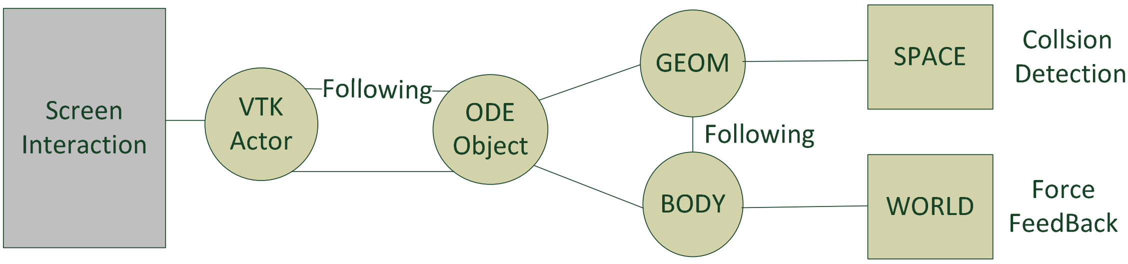


Figure 4.12 Relationships between VTK and ODE

In the first impression, this model is perfect for the interaction with the ODE model. However, when we try to realize the whole process, there are two loops in the flow chart (Figure 4.13) that have to be executed simultaneously. Loop1 is the *interaction with the model* performed with VTK. Loop2 is the *collision detection* performed with ODE.

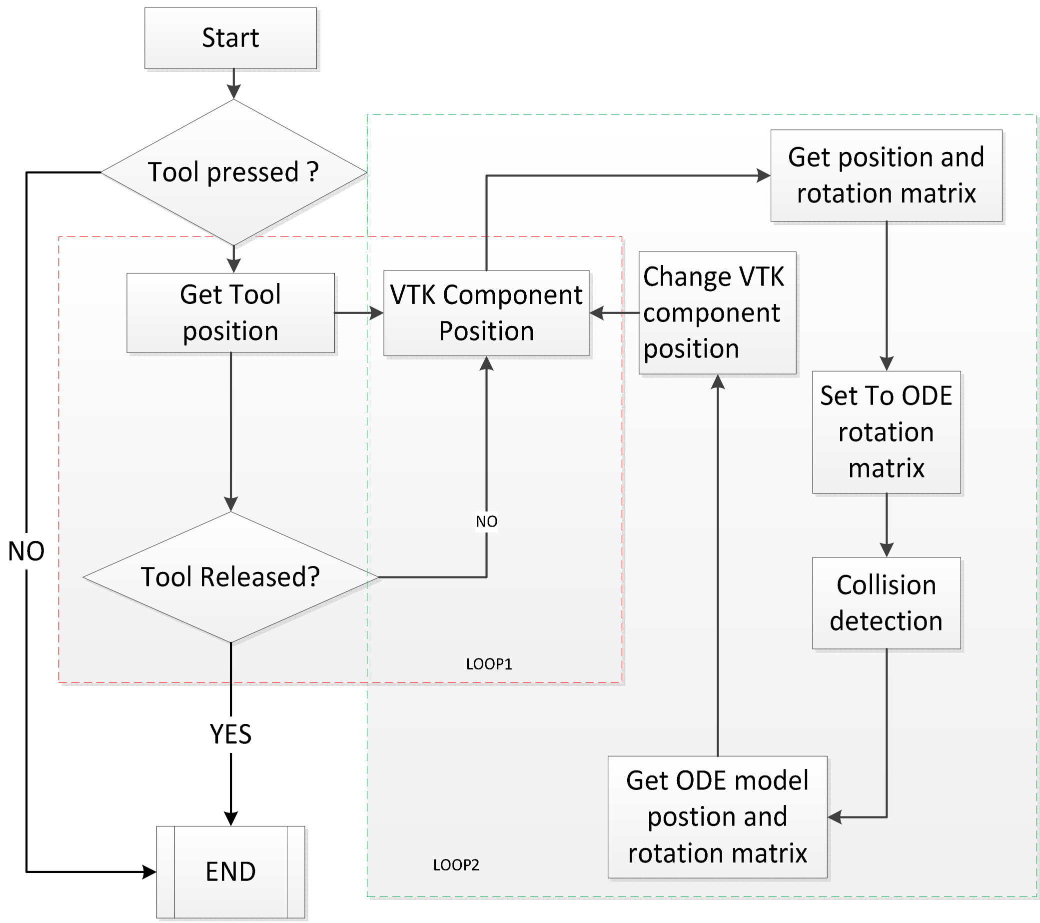


Figure 4.13 Flow Chart for Disassembly collision detection.

In the disassembly process, these two loops affect the position of the parts' models created by VTK. Two facts have to be considered during the disassembly simulation. The first one is that all the assembly parts components are connected to each other and the distances among them are zero (without gaps). So, during collision detection, if the distance is zero, all the parts are supposed to collide in the initial phase and the algorithm for collision detection will not be able to run. Sometimes it is even possible to cause the system crash.

The other fact is when the target part is colliding with other components. In this case the target has to change its original moving direction due to the feedback force. However, the other components of the assembly will remain in their initial position because of the friction.

In order to detect collisions, two methods are applied. The first one is the so called *space collision detection* algorithm. It consists in detecting the collision between parts in the different ODE spaces and ignoring the collision if the models belong to the same *ODE space*. When a *3D* assembly is imported in the ODE (WRL format), all its parts are put in the same ODE space. Then, the collision detection is performed only between components belonging to different ODE spaces. When a component is disassembled, its ODE space changes, then space collision detection algorithm is called by clicking on ODE button on the disassembly simulation interface (see Section 4.4.5).

The other method for collision detection is using the so called *Kinematic criteria of ODE mass method*. For every ODE body, there is one mass associated to it. If the mass of the ODE Kinematic is active, the associated model is too heavy to be moved by the collision detection force feedback. Thus, the method can be used to simulate the unmovable characters of the components because of the friction influence (see Section 4.4.5).

4.4.3 Stereo Rendering

There are some techniques allowing simulating 3D graphic on a 2D display device such as: using perspective and scale, shading to confer depth, motion/animation to see all the sides and so on. However, the most effective way is binocular parallax, which is a result of viewing 3D objects with our two eyes. Since each eye receives a slightly different view (Fig.4.14), our mind interprets these differences to determine the depth of the picture. Most 3D movies take advantage this principle to realize the 3D vision (wearing special glasses when watching the movie). In the evaluation of sequences of disassembly, this effect can be valuable to provide the real disassembly environment for product disassembly evaluation and the stereo viewing can help in determining the relative positions of each component. In order to generate correct left and right eye view differences, the proper method for rendering the binocular parallax is the key.

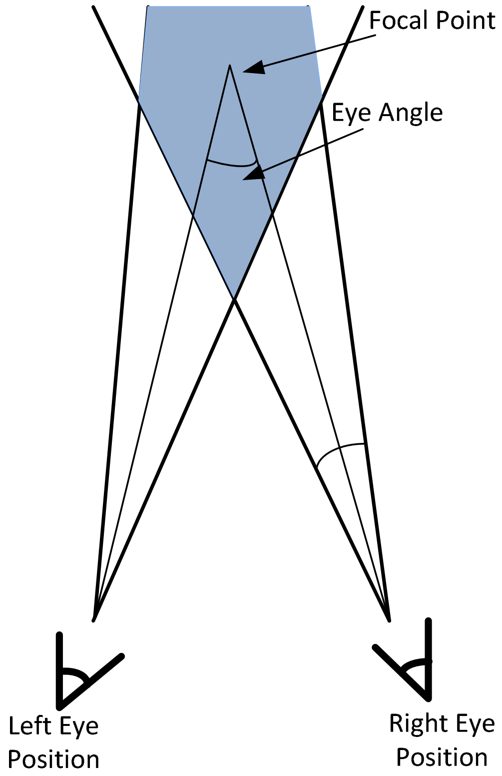


Figure 4.14 . Binocular parallax technique for stereo rendering

Most methods for Stereo Rendering are based on: *time multiplexed* or *time parallel* techniques. Time multiplexed techniques relay on the alternating images. When they are viewed with both eyes, they appear as one image that keeping jumping from left to right. A special glass is designed so that each lens consists for a liquid crystal shutter that can either be transparent or opaque. It makes sure the left eye image is being displayed, the user's left eye can see and similarly for the right eye. This method requires viewing images on a television, not the monitor connected to computer. Time parallel techniques can display the images of two eyes in the same time. The two separate screens are generated for each of the eye. To generate the two video streams, the technique needs either two graphic cards or one cart able to generate separated outputs. The biggest disadvantage to this approach is the cost of the hardware required. There are still two other technologies for stereo rendering implemented using the above two techniques. The first one is red-bleu (red-green or red-cyan) stereo which is requiring to wear glasses that filter the entering light. Left eye can only see the image a red filter and the right through a blue filter. The benefits for this method are that all the images can be displayed on a monitor, paper or film, and all one needs to view them is an inexpensive pair of glass. The second technique is to separate the different views by using polarized lights. In our application, we have chosen the techniques for stereo rendering using VTK library. Time multiplexed techniques images are shown as in Fig. 4.15.

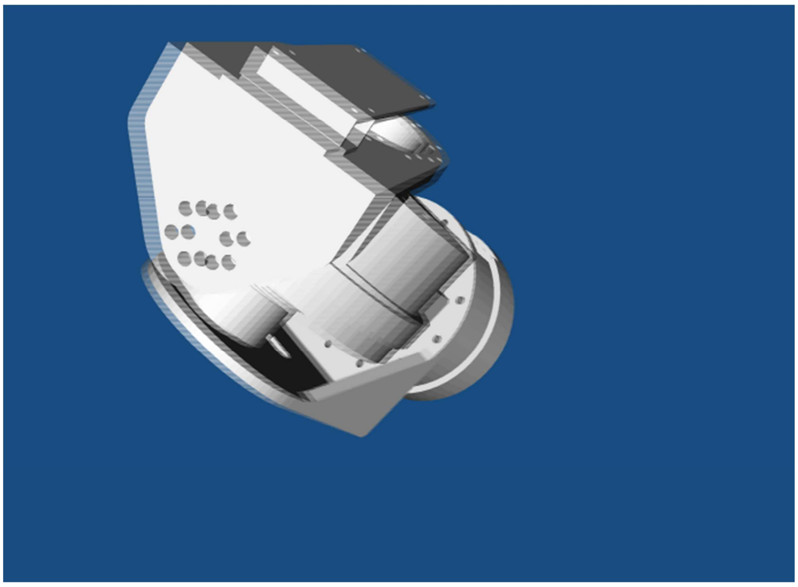


Figure 4.15 Stereo Rendering.

4.4.4 Force feedback and Virtuose 6D35- 45

In order to perform the disassembly operation simulations we choose Virtuose 6D35-45. It is a six degrees-of-freedom (DOF) haptic device, specifically designed for application in VR environment. It is especially recommended for *scale 1* manipulation of virtual objects such as assembly/disassembly simulation, ergonomic studies or maintenance training. Modular in design, it can be purchased as a 3-DOF device, and later upgraded to 6-DOF. The main characteristics of the used Virtuose we uszd, available in GINOVA Platform, Grenoble INP are:

- Workspace: 450 mm
- Maximum force: 35 N
- Continual force: 10 N
- Maximum torque: 3 Nm
- Continuous torque: 1 Nm

The original library is on C++. Then C++ library is changed to Python. And finally Python is used to connect with the Virtuose 6D35- 45 arm.

4.4.5 The whole VRDE environment

The whole system is based on Python language. (part of the code is presented in Appendix A). The outputs are *3D* sound and stereo displays. The interface is developed based on Visualization Toolkit (*VTK*) library. We provide *VTK* library for creation and interaction with the *3D* models. In order to prevent interfering paths generation, the real time collision detection is developed based on the *ODE* (Open Dynamics Engine library). As previously said, *ODE* is particularly adapted for simulating moving objects in VR environments thanks to its advantages, namely: robustness and stability. At the same time the user may change the structure of the system in real time.

The developed software can support *WRL* and *STL* format files. In the example presented here below, the Solidworks models were imported in the application in *STL* format (Fig.4.16). In order to count the pixel for visual score (*VS)* calculation, the target's color (here in red) should be different from the other components (details are presented in Chapter 5).

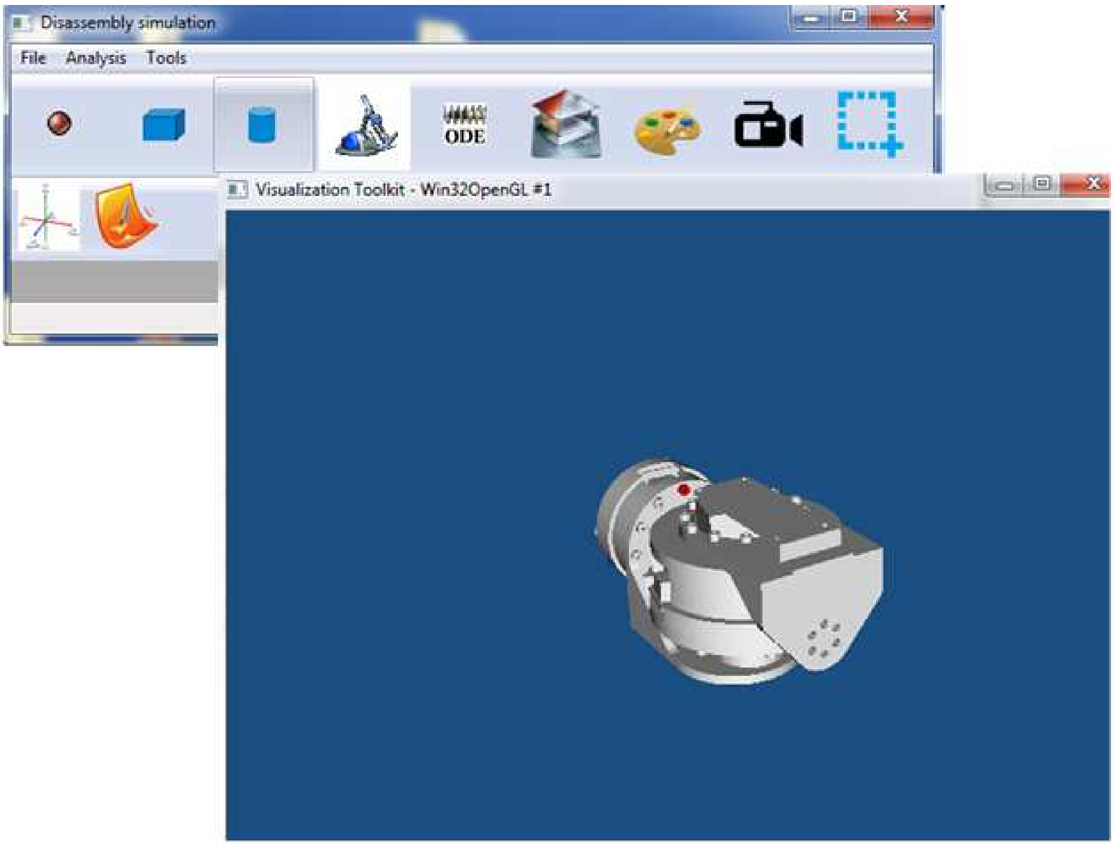


Figure 4.16 Virtual platform Interface

4.5 Conclusion

In this chapter, the related technique of 3D graphic pipeline and a new application for virtual simulation based on *Python* programming language associated with *VTK* and *ODE* libraries were introduced. The related device for the performed experiments and the collision detection algorithm was also introduced. The application is the principle software used for performing disassembly operation simulations and two examples for disassembly simulation are presented in Chapter 5. At this stage we can say, that the software can be naturally adopted by a variety of virtual environment applications for A/D sequences evaluation.

Chapter 5
Method for disassembly operations' efficiency evaluation. Integration in Virtual reality environment

This chapter presents a method for evaluation of disassembly sequences. The design of a virtual environment and the implementation of a computer application that supports the evaluation of disassembly sequences are presented as well. The main objective of such application is to help designers analyzing the difficulty of disassembly operation execution in a virtual reality environment (VRE). For this purpose seven criteria, divided in two categories: for ergonomical and traditional processing evaluation are proposed. The criteria are presented by dimensionless coefficients automatically calculated by the realized application thus allowing evaluating disassembly sequences.

5.1 Introduction

As discussed in Chapter 3, for disassembly it is important to eliminate the components which are unrelated to the target components prior to sequence generation. Considering the least level of disassembly graph method for generating all the possible sequences, how to choose one with the least cost value in the real disassembly process is still an issue. Thus, the evaluation method within the virtual environment to value the disassembly sequences is proposed in this chapter. The proposed method for disassembly operation evaluation deals with the following two aspects:

- Considering Ergonomic parameters in virtual reality environment (VRE).

In the Design for manufacturability (DFM) principle and in particular in the Design for Disassembly (DFD), an operator is often involved in order to test two concerns namely: his/her posture and the visibility of the disassembly parts. One effective and common method for ergonomic evaluation of A/D operations is using a digital mock-up (DMU) in a VRE. DMU is a realistic computer simulation of a product containing all the required functionalities for design, manufacturing and product service environment. The methods for ergonomic evaluation in virtual environment (VE) often involve a human model.

At the same time, those methods are relatively costly and time consuming. Thus, their mainly application is limited in the expensive products' development fields. This is the reason that lots of developments involved in VR with human models are limited mainly in big industries such as: automotive and aerospace. In order to address this limitation of using the human DMU, a more simplified method is proposed here allowing solving the disassembly evaluation in VR environment. In this chapter, we provide a new way to evaluate the difficulty to perform disassembly operation sequences in virtual environment instead.

- Considering the traditional disassembly procedure evaluation method.

The traditional disassembly procedure evaluation is using a cost function presented in many works (for more details please referring to Chapter 2). The majority of disassembly evaluation research focuses on some criteria related to the disassembly process in manufacturing industry such as: the number of parts involved, the tools changes times, the stability of sub-assembly, the fixtures etc.

5.2 Method for disassembly operations evaluation

In the proposed method, we aim at dealing with various criteria related with disassembly operation evaluation. The objective is to develop a VR based system enabling interactive analysis and evaluation of disassembly operation by considering the proposed Ergonomic Geometric Removability of the components and the traditional processing evaluation. Instead of ergonomics simulation with a human model, it introduces some new parameters such as: visibility, neck and bending scores, amongst others, thus allowing performing and evaluating disassembling task in a VR environment.

5.2.1 Ergonomic Auto Evaluation method

The purpose of disassembly Evaluation is to obtain approximate disassembly time for a product by using formulas derived from the information pertaining to connect parts instead of disassembling the product in reality. As we mentioned in Chapter 2 (section 2.4), the purpose of Ergonomic engineering is trying to fit the task to the human and not the human to the task where the key point for an effective application is to gain a balance between the human body characters and the task demands. Thus, in this chapter we propose a method for disassembly evaluation in VR environment. Instead of focusing on the authenticity assessment by comparing the results of VR and real task in reality, the proposed Geometric Removability Analysis method is focusing on the evaluation of disassembly difficulty in VE which consists in:

- Analysing the Physical position of the operator.

In order to address the Geometric Removability Analysis of disassembly, first a study should be done on the physical position analysis when the operator disassembles the product in the VE. For this purpose four geometrical parameters related with the human operation convenient in the VR environment are proposed (Fig. 5.1):

- the first parameter is angle c_1 between the visual direction and the vertical direction, (less than 90°), if >90°, the operator have to rise his/here neck in order to carry out the task. The visual direction defines the eyes direction in the VE,

- the second parameter is angle c_2, between the visual direction and the component moving direction (around 45°, if more than 90°, the component cannot be operated properly),

- the two other parameters are the horizontal distance d_1 between the operator's position and the center of component, and the distance d_2 between the operator’s eye and the center of the component. They measure whether the operator needs to bend over for completing the operation. For a disassembly operation, the least value of d_1 is normally fixed by the workspace itself or the fixture. For better visibility d_2 must be shorter. In this case the operator needs to bend over for completing the task.

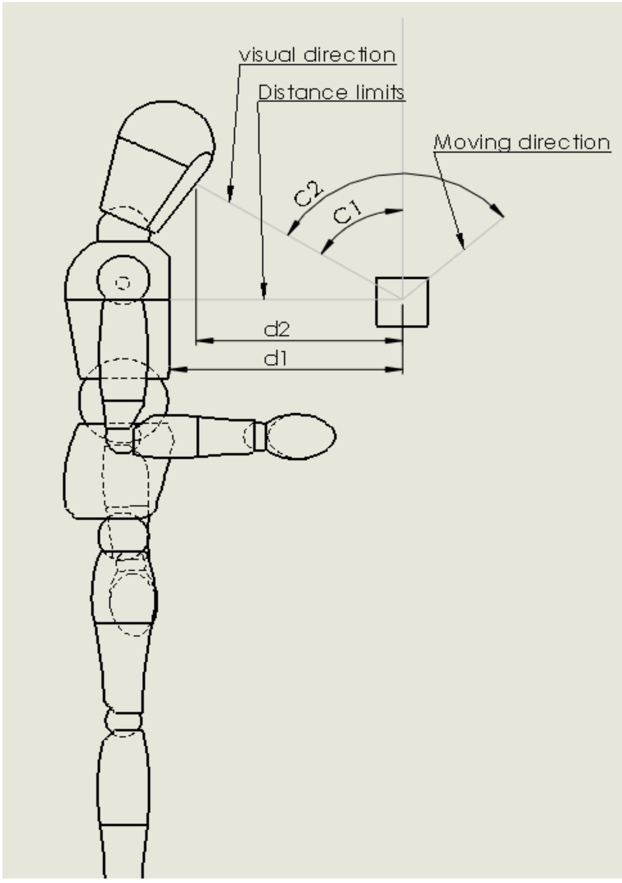


Figure 5.1 Four geometrical parameters related with the human operation.

- Replacing the human eyes by a camera.

Its principle idea consists in using a camera to replace the operator for automatic estimation of the ergonomic parameters. However, this method does not consider the VR environments and the interaction operation during the disassembly process. Here, we propose using a camera to replace the 3D human model and in particular the eyes of the operator. Then the analysis of the distance and angle related with the component disassembly operation direction, and the component position in the VE is used for the removability evaluation by considering the proposed ergonomic parameters (Fig.5.2).

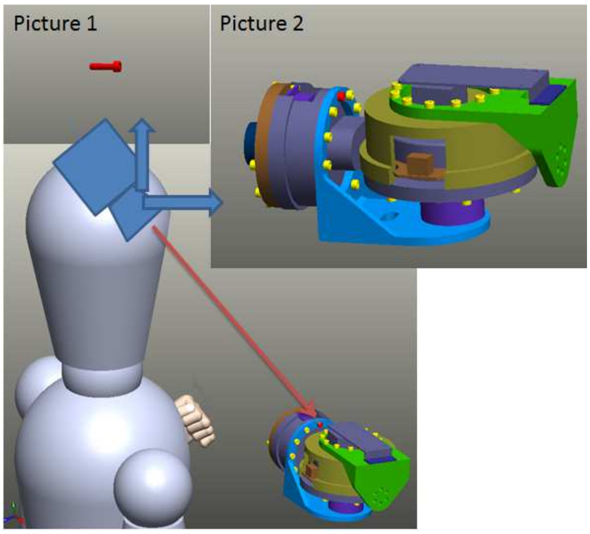


Figure 5.2 Camera as the eyes of the operator.

- Ergonomic Auto evaluation

  Three criteria for ergonomic disassembly evaluation are proposed:

  i) visibility score (VS)

Replacing the human eyes by a camera has limitations as the human physics is not taken into account. In order to address the visibility score, the method we propose have to consider the human physics as well. Thus, in the process of operation, the initial position of the camera should be the eyes of the operator by considering his/her height (Fig 5.2).

In order to calculate the visibility score for a bolt for example (Fig.5.3 and Fig. 5.5a), firstly, the camera should be in the direction and the position of the human eyes. In this way, there are two images taken by a camera. One is the bolt itself noted by red pixels. The other image is the bolt in the assembly surroundings. The color pixels, here in red, stand for the visibility of the target part. Then, the numbers of pixels in the two images are counted automatically in time using open CV library.

Thus, the operations' ranges should be limited in the movements of human's head and body.

In the disassembly model, all previously disassembled parts are displayed in one image taken by the camera. The color pixels, in red in the sub-assembly (Fig 5.4.a) stand for the visibility of the target part. In another image, only the disassembly part is displayed (the other parts are hidden) which shows the maximum visibility for this part (Fig 5.4.b).

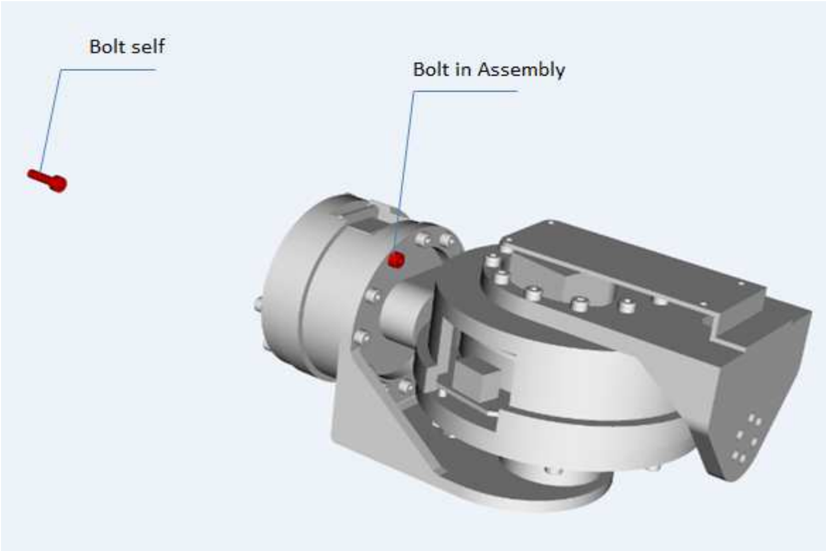


Figure 5.3 Visibility for a bolt.

It must be noted that, for these two images, the distances between the camera and the part should be the same. Then, the number of pixels in the two images is automatically counted. In this way, the ratio of red pixels between the visible portion (Fig5.4.a) and the whole target part (Fig 5.4.b) is used for measuring the visibility of the concerning target part.

(a) Target part in the Sub-assembly (b) Target part

Figure 5.4 Calculation of the visibility score (red highlighted areas).

Thus, the proposed visibility score v is defined as the ratio between the number of red colored pixels in the current image v_a of the target part (Fig. 4.a) and the number of red colored pixels of its whole image v_b (Fig. 5.4.b) captured by the camera:

$$v = \frac{v_a}{v_b} \tag{5-1}$$

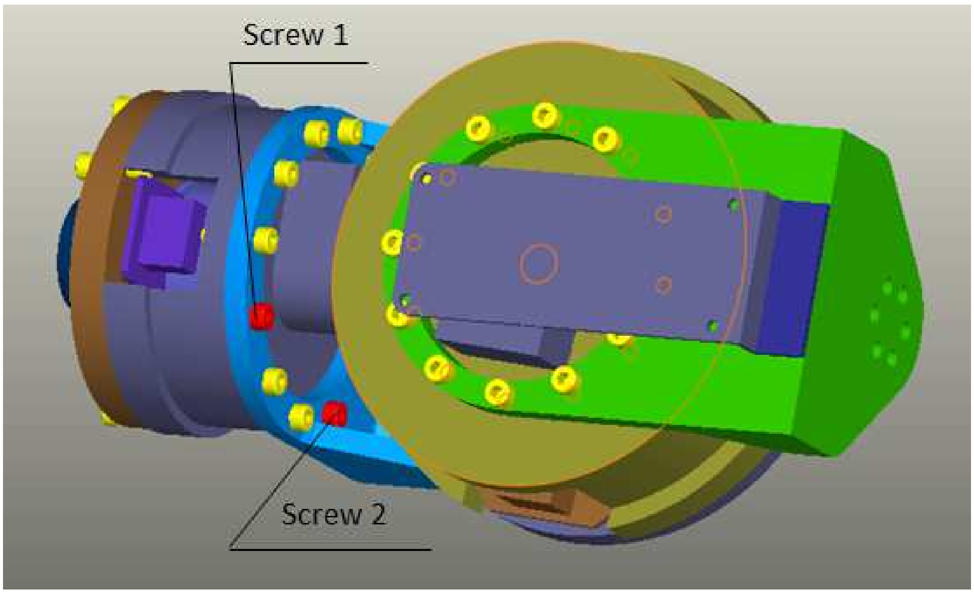
with $v_b \neq 0$.

If there is no obstacle part to hide the target, the visibility score is 1. If the target part is completely hidden by other parts, the visibility score is 0. Thus, the average visibility for the disassembly sequence is:

$$V = \frac{1}{m}\sum_{i=0}^{m} v_i \qquad (5\text{-}2)$$

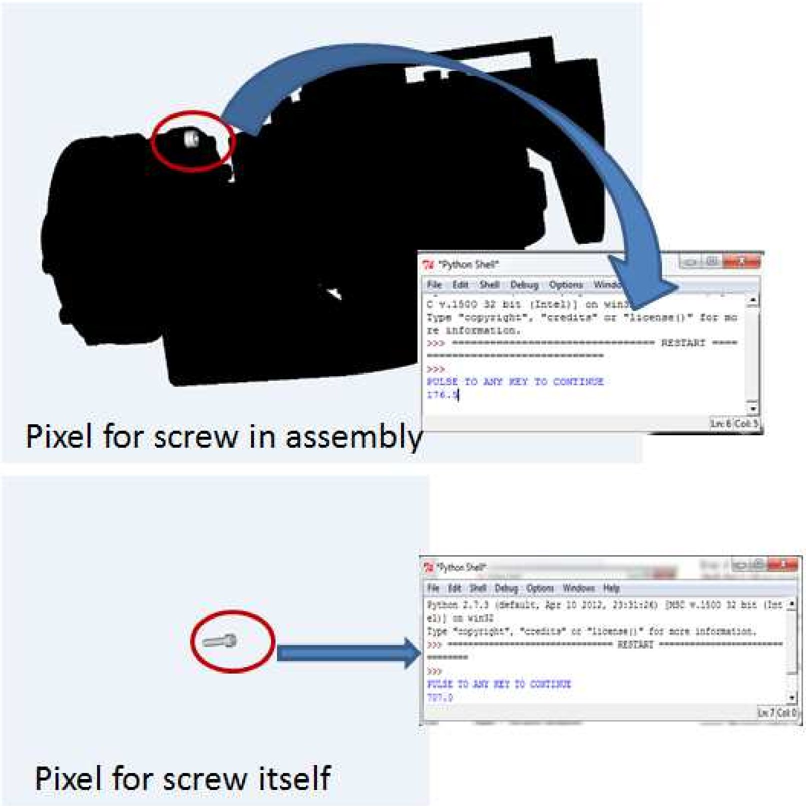
where: m is the number of components in the assembly.

The pixel counting is based on the OpenCV library (http://opencv.org/). In order to calculate the visibility score and the pixels of the target, its color (here in red) must be different from the other components in the assembly. For this purpose the other components are becoming black colored in grey scale as shown in Fig. 5.5b.



a) Mechanical assembly with disassembly targets.

As a result, visibility scores v for the two targets are: Screw 1: v=0.249646393211 and for Screw 2: v=0.168912236542. Therefore, for human operation Screw 2 is more difficult to be disassembled in the VR environment as its score is smaller than the score of screw 1.



b) . Pixels for screw

Figure 5.5 Pixel calculation for target components.

ii) Neck score (NS)

Two types of Neck Score are usually used for ergonomic evaluation: component heads and text heads. Here, we use Rapid Upper Limb Assessment (RULA) algorithm proposed by McAtamney and Corlett [McA 93] in order to evaluate the exposure of workers to risk of upper limb disorders.

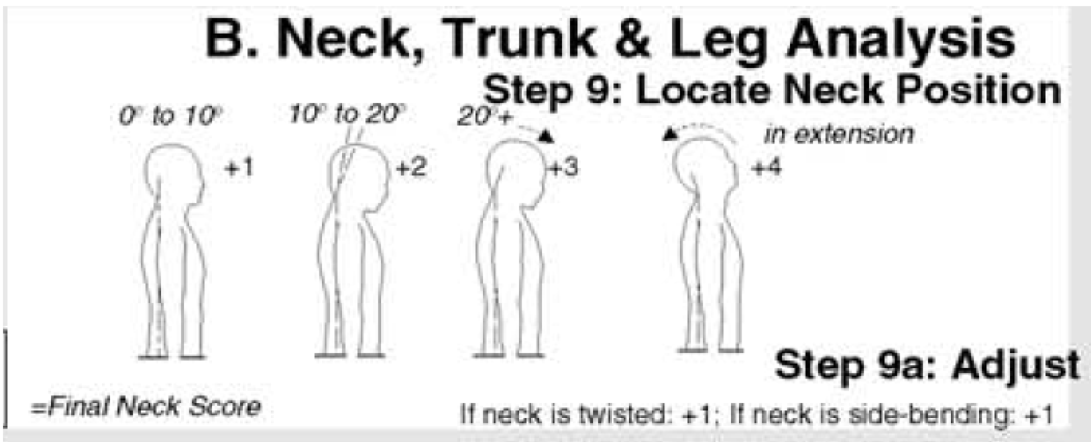


Figure 5.6 Neck part from RULA sheet [McA 93].

Neck score (NS) measures lateral and forward rotation angles of the neck. There are two angles susceptible to affect the neck fatigue.

The first one is the forward rotation angle c_1 (Fig. 5.1). The forward score *F* we propose is:

$$F = 1 - 9\frac{\pi - 2 * c_1}{2\pi} \tag{5-3}$$

According to RULA sheet (Fig. 5.6), if angle c_1 is more than 90° or less than 70°, the forward score is 0 (zero).

The other angle is the lateral rotation angle c_3 of the neck as shown in Fig 5.7. If the value of c_3 is between 0° and 20°, the lateral rotation score *s* we propose is:

$$s = 1 - 9c_3/\pi \tag{5-4}$$

Thus the average Neck score NS we propose is:

$$NS = \frac{1}{2}(f + s) = 1 - \frac{9(c_1 + c_3)}{2\pi} \tag{5-5}$$

In the realized application, we consider that if the value of c_3 is more than 20°, the side scores for lateral rotation is s=0, which implies that the side bending is too big for the operator.

Finally, the total neck score NS can be calculated as:

$$NS = \begin{cases} 0 & (c_1, c_3 > \pi/9) \\ N = \frac{1}{2}(f + s) = 1 - \frac{9(c_1 + c_3)}{2\pi} & (c_1, c_3 \leq \frac{\pi}{9}) \end{cases} \tag{5-6}$$

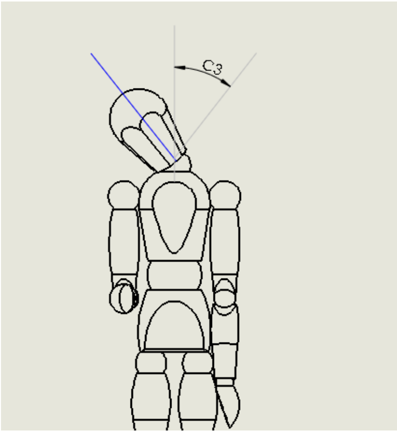


Figure 5.7 Neck lateral rotation.

iii) Bending score (*BS*)

Another parameter which is affecting the ergonomy of the disassembly operation is the bending score (*BS*). Its value is calculated from the trunk bending angle as shown in Fig.5.8.

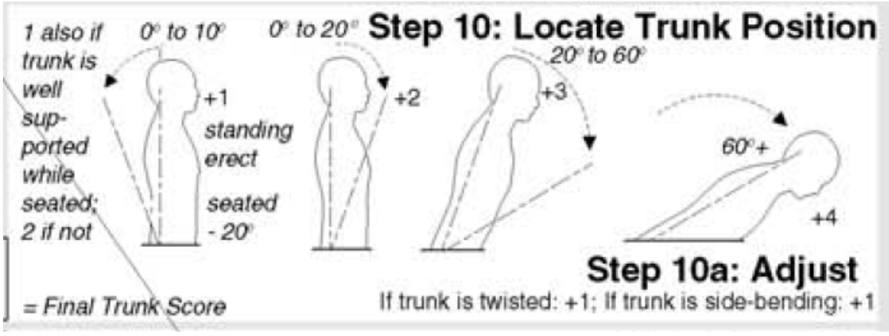


Figure 5.8 Bend over reference from RULA sheet [McA 93].

If angle c_2 ranges from 0° to 60°, *BS* is defined as:

$$BS = 1 - 6c_2/\pi \qquad (5\text{-}7)$$

Note that in the worst case (c_2>60°), the bending score is 0.

The three score (VS, NS, BS), proposed here above, formulate a strategy to create a simple analysis for ergonomics evaluation. However, the problem is how to use this approach in the absence of 3D human model. For this purpose, as previously said, the proposed method consists in replacing the human model by a camera. The latter is used to detect all the angles and distance necessary to calculate the overall score of the proposed three ergonomic criteria.

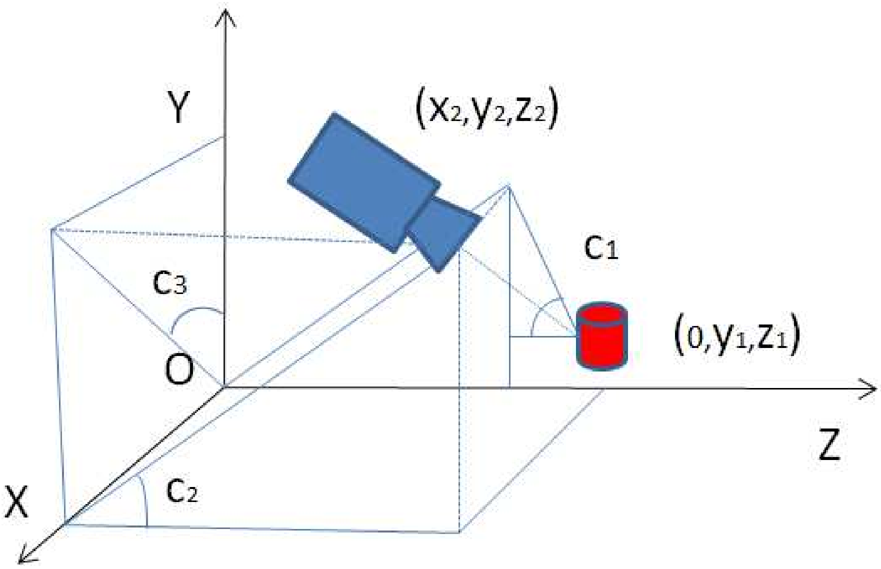


Figure 5.9 Ergonomic angles and Camera position relationship.

The proposed procedure using a camera for ergonomic evaluation consists in:

- Define the work environment.

First the target component is set in the OYZ plane (Fig 5.9). Then, the human operation plane is defined as the parallel plane with OYZ in positive x direction.

- Define the position of the camera.

According to the workspace and position of the target component, define the position of the camera. Note that, the initial position should consider the operator height (size) and the real distance between the operator and the camera. For example, distances d_1 and d_2 (Fig.5.1) should not be too small. Because we use camera, instead of human body, the suitable position for the camera is not known. Consequently it should be defined by the operator before the beginning of the disassembly operation.

- Use the camera to detect the geometrical parameters namely: distances d_1, d_2 and angles c_1, c_2 and c_3.

$$\tan(\mathrm{c}1) = \frac{y_2 - y_1}{z_2 - z_1} \qquad (5\text{-}8)$$

$$\tan(\mathrm{c}2) = \frac{y_2}{z_2} \qquad (5\text{-}9)$$

$$\tan(\mathrm{c}3) = \frac{y_2}{x_2} \qquad (5\text{-}10)$$

Then according to formulas (5-1), (5-5) and (5-7) the overall score OS for the ergonomic evaluation of disassembly operation is:

$$OS = VS + NS + BS \qquad (5\text{-}11)$$

It considers in the same time the ergonomic parameters of the operation environment and the visibility of the components. Note, that they are closely related. For instance, let us only consider the human comfort. If the visibility score is low, the operation will be difficult to realize even with high ergonomic score and vice-versa.

- Example of ergonomic disassembly operation evaluation

In order to demonstrate and validate the proposed method an example is presented here below. The case study involved a portion of bolts disassembly operation in the created VR environment. The original operation using 3D human model is shown as in the Fig.5.10. Instead, in order to avoid using of 3D human model, a camera is applied which replaces the eyes of the 3D human model as shown in (Fig.5.11). Experiment consisted on virtual disassembling two screws from a mechanical assembly (Fig. 5.5a).

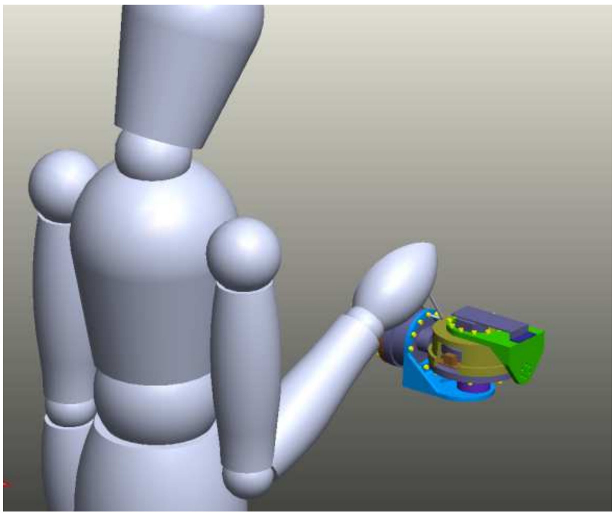


Figure 5.10 Disassembly operation case study.

As previously said, the initial position of the camera should be the eyes of the operator by considering his/her real height. The activities' ranges should be limited in the human head and body's movable ranges. Note, that this is a little awkward in the scene of the VRE. This is because, in general, the camera has to observe the objects, and can be moved anywhere if the operator wants to. However, in our application, the movement of the camera is restricted in consideration of the human body dimensions.

The mechanical assembly is imported from a CAD system in WRL formats.

In order to prepare the pixel detection for the target components, after importing, the color of the other components (except the targets) are set into the same grey color as shown in the Fig.5.5.a.

As presented here above (see Fig. 5.9), the positions of the camera and the object are first build. Note, that the position of the camera is related with the human height (here 175cm).

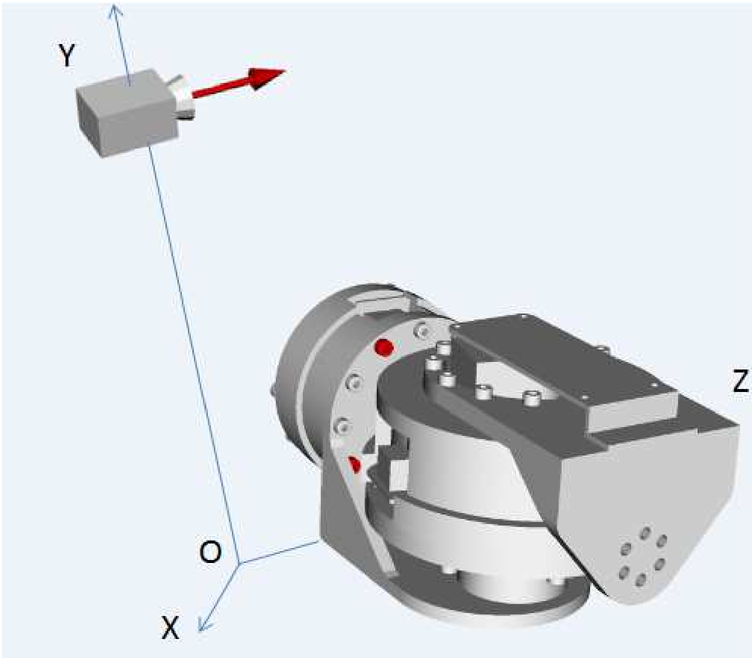


Figure 5.11 Original positions of the camera and the targets.

Then, the operator may remove or rotate the camera in the convenient position for observation. When the target is selected, its pixel of image and position are record automatically for later analyses (Rq. The cursor of the tool is disappearing first in order to save the image pixels).

Then, angles c_1, c_2 and c_3 are calculated according to the position values of the camera and the targets (screws), by equations (5-8), (5-9) and (5-10) respectively.

And finally get the overall score OS for the operation difficulty evaluation by eq. (5-11).

Let us note that, the values for visibility of a part depend on the way that the operator is handling the components in the VRE. Two screws disassembly operation were involved in the performed experiments for disassembly simulations. According to the proposed method for disassembly operation evaluation (formulas for the three scores) the results for overall score (*OS*) for each disassembly operation (screw 1 and 2) are showing in Table 5.1.

It is seen that screw2 is more painful for the operator neck as NS of Screw 2 is less than NS of Screw 1. Concerning bending score BS of Screw 2 is smaller than BS of Screw 1 which means that the operator needs to bend over more for disassembling screw 2.

With regard to visibility score (VS); screw 2 is more difficulty to be seen compared with screw 1 as its VS is smaller than screw 1.

In conclusion, the overall score of Screw 1 is bigger than Screw 2 which indicates that it will be easier to be disassembled in ergonomic point of view.

Table 5.1 Overall score for screws disassembly operation

Operations	**Geometric Removability Analysis**				
	Visibility score VS	***Neck score NS***	***Bending score BS***	***Overall score OS***	***Evaluation***
Screw1	0.654334	0.718317	1.0	2,372651	Easy
Screw2	0.547912	0.369079	0.832401	1,749392	Diffcult

5.2.2 Traditional processing evaluation method

The traditional called also processing disassembly evaluation procedure, instead of considering the ergonomic evaluation, considers some criteria related with the technological conditions for disassembly process execution. Thus, we propose four new parameters for disassembly evaluation presented by dimensionless criteria which are:

- Stability of sub-assembly.

Unlike assembly operations, the stability of the sub-assembly is an important property for the disassembly operations evaluation. The sub-assembly is defined as the remaining parts of an assembly (mechanism) after removing the current target part. Thus, sub-assembly stability is defined as the possibility of the remaining parts to be in steady state when a part is taken away from the assembly. For unstable sub-assembly disassembling, some extra fixtures and tools must be involved; otherwise the operation will be dangerous for the operator. For this purpose gravity is implemented in the proposed method in order to simulate the real gravitational environment. Thus, the stability score Sta of the sub-assembly is defined by:

$$Sta = 1 - \frac{f}{m} \tag{5-12}$$

where: f is the number of the components falling, down in the gravitational field, calculated by the developed software. The value of Sta ranges from 0 to 1. For f=0, the stability is maximum, consequently Sta=1. The worst situation is for f=m, when Sta=0.

- Number of tools' changes

During disassembly operation the number of tools' changes is an important factor for the operation time estimation. For a product with m components, the worst situation for the number of tools' changes (n) is when n=m-1. It means that for the disassembly of each part, the tool has to be changed independently on the number of parts. The dimensionless coefficient of tools' changing T is defined as:

$$T = 1 - \frac{n}{m-1} \qquad (5\text{-}13)$$

where $m \geq 2$ and $n \geq 0$. The value of T ranges from 0 to 1. Obviously, the best situation is when it is not necessary to change the tool (n=0) to disassemble the components. This is the ideal situation and T=1. If n=m-1, as mentioned here above, this is the worst situation with T=0.

- *Set of directions for removal (SDR)*

As it was said in Chapter 3 (Section 3.2.2) the basic idea here is to use the contact surfaces of the components in order to determine the required set of directions for removal (SDR) (Pom 04). SDR represents the possible separation directions of a component with regard to its surrounding components. To remove the target component from a product, each of its contacts has to be identified in order to get the possible SDR. Concerning the contact identification for A/D simulation we used the method of Iacob (Iac 08) based on the analysis of the functional surfaces of the parts.

For a component moving in 3D environment with 6 DOF, the disassembly directions are in the $4\pi r^2$ surface where r is the radius of the sphere (Fig. 5-12). This surface is the image of 360° volume angle. Therefore, for any SDR, the disassembly surface for a component is:

$$s(\phi, \theta) = \int_0^{\phi} \int_0^{\theta} r^2 \sin\theta d\theta d\phi \qquad (5\text{-}14)$$

where: θ and ϕ are polar and azimuthal angles respectively.

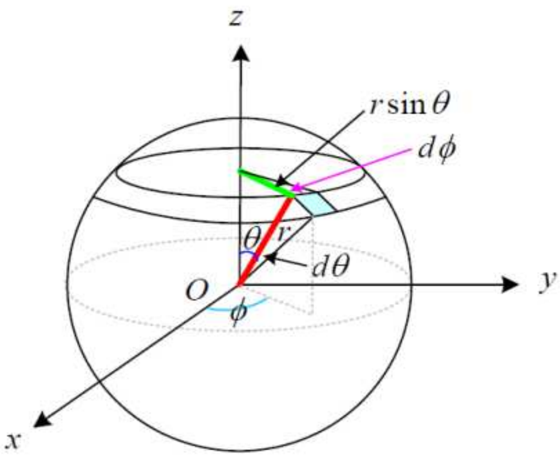


Figure 5.12 Surface representation of the disassembly angle.

Consequently, the relative score for the disassembly angle C for a component can be calculated as the ratio between the disassembly surface angle and the whole surface of the sphere:

$$C = \frac{s(\phi,\theta)}{4\pi r^2} = \frac{1}{4\pi}\int_0^{\phi}\int_0^{\theta}\sin\theta d\theta d\phi \qquad (5\text{-}15)$$

We consider that *C* is the image of SDR in 3D space. The value of C ranges from 0 to 1. The best situation is when C=1 (all the possible movements are feasible) and the worst one when C=0 (there are no possible movements).

- Changes of the path orientation

Another essential criterion to estimate the difficulty to disassemble a part is the changes of path orientation. Let us consider a path and a number of points A, B, C on it situated in equal distance (step) u mm chosen by the operator (Fig 5-13). At each point, a tangent vector on the path (curve) is defined. The first one is called referent vector, situated in the beginning of the curve (here in point A).

Then, the angle α between the referent vector and the next tangent vector, called local vector, (here at point B) is calculated. If α is smaller than a limit, imposed by the operator, for example $\alpha << \pi/3$, it is considered that there is no Path orientation changing. Then angle α between the tangent local vector in point C and the referent vector is calculated. If α is bigger than the limit, it is considered that the direction has changed. In this case, the local vector (here in C) becomes the referent vector and so on.

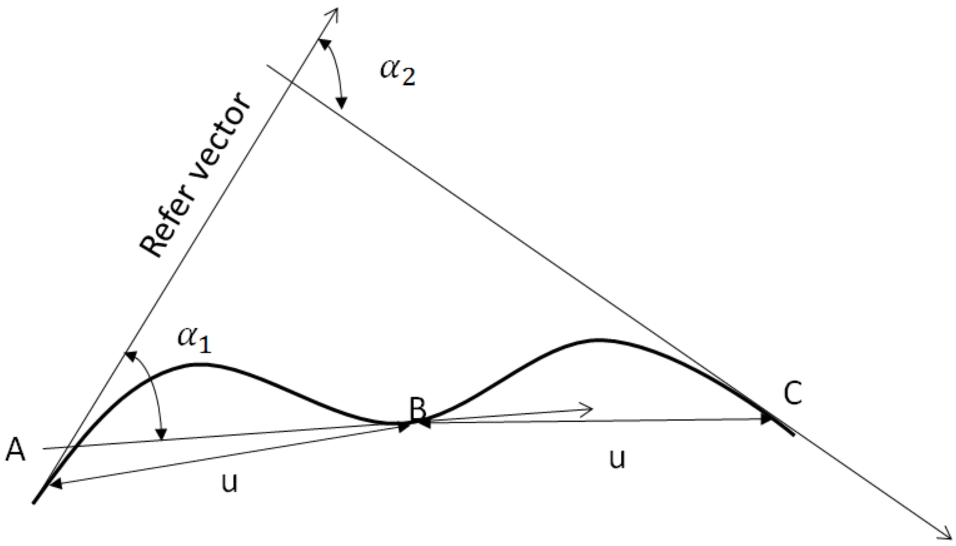


Figure 5.13 Path orientation changing.

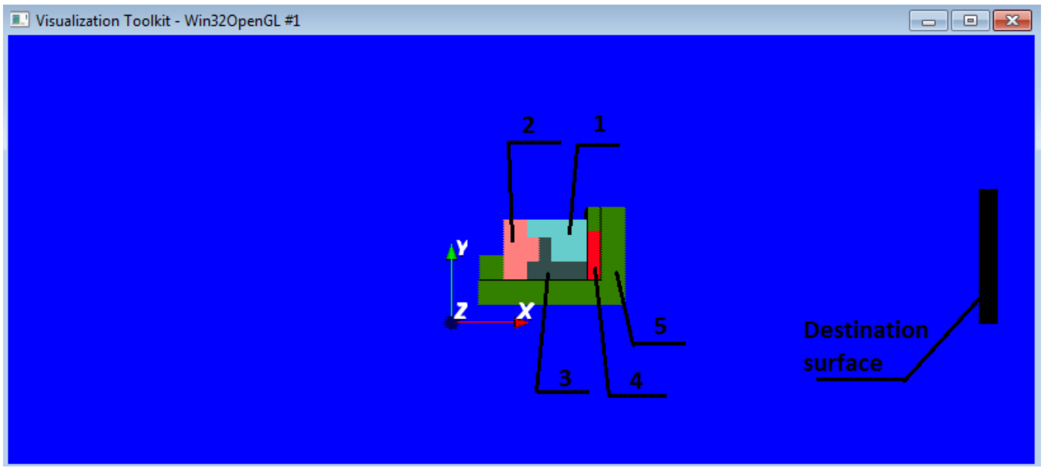
A dimensionless coefficient P is proposed allowing estimating how many times the path changes its direction orientation. Thus, the proposed path orientation changing P is:

$$P = \frac{1}{t+1}\sum_{i=0}^{t}\frac{\left(1-\frac{\alpha_i}{\pi}\right)}{i+1} \tag{5-16}$$

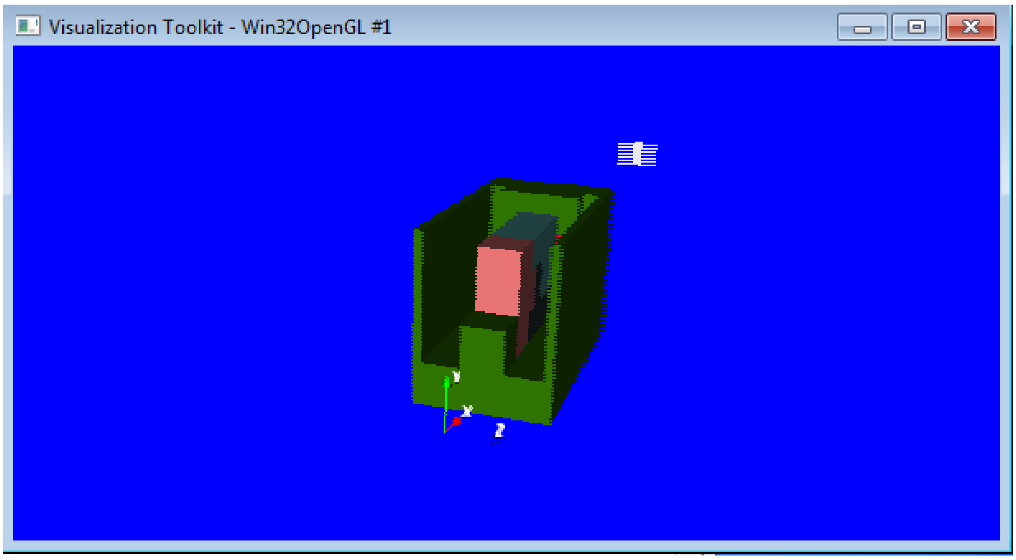
where: t is the number of times for orientation changing. The value of P ranges from 0 to 1. The ideal path is when t=0, $\alpha_i = 0$, the path is a straight line, and consequently P=1. For $\alpha_i = \pi$, P=0, which is the worst situation. The four criteria presented here above by dimensionless coefficient are integrated in a Virtual reality disassembly environment (VRDE), thus allowing to evaluate the disassembly sequences' complexity.

5.3 Implementation and results

An application for disassembly simulation was developed running on the proposed virtual reality disassembly environment VRDE (Chapter 4). Here below it is illustrated by an example of a five-parts mechanism (mechanical assembly) disassembling (Fig. 5-14). The disassembly experiment consists in moving all the parts from the mechanical assembly to the destination vertical surface (wall) as shown in Fig. 5-14a. As said in Chapter 4, the collision detection is performed with ODE. Note, that if a collision happens, the collision force changes the moving direction of the VTK model.



a) Cross section front view of the mechanical assembly



b) 3D stereoscopic view

Figure 5.14 Assembly view in virtual reality environment

5.3.1 Simulation process

The process for disassembly simulation evaluation consists in two main steps, namely: *operation* and *calculation*.

(1) Operation (manipulation) of the camera: As previously said, the operator removes or rotates the camera in a convenient position for observation. As presented in the *Visibility score* paragraph and the *Example of ergonomic disassembly operation evaluation* of Section 5.2.1, the environment coordinates for the camera position and the object position related to the human height (*175cm*) are first built. We call this *the operation step* of the process.

(2) Calculation of the proposed four criteria for technological (tradition processing), disassembly evaluation namely: *disassembly angles* or *Set of directions for removal* (SDR),

stability of sub-assembly, *time of tools' changes* and *path orientation changing* by calculating the proposed dimensionless coefficients. We call this the *calculation stem* of the process.

i). Disassembly angle: *Set of directions for removal* (SDR),

First, SDR is calculated prior to disassembly operation simulation. It consists in detecting polar and azimuthal angles according to the assembly relationships amongst the corposants. Thus, the value of *C* can be gotten in real time by applying eq. (*5.14*)

ii). Stability of the sub-assembly

Concerning the stability detection, note that component *5* (Fig. 5-14), being the base component, is not concerned by falling down under the effect of the gravity. After disassembling components *1* and *2*, if component *5* is the auxiliary target, components *3* and *4* will be in unstable state. In this case, to continue the simulation, additional fixtures for components *3* and *4* have to be added in order to ensure the stability of the sub-assembly. If a fixture is necessary to be added to a component, the assembly time will increase. For this reason, in the realized VR for disassembly sequences' evaluation, a punishing time for this component is allocated by the operator.

iii). Number of tools' changes

In order to evaluate the criterion number of tools' changes, it is assumed that for disassembling components *2* and *3*, the same tool is used. However, for disassembling the other three components, three different tools are used for each of them.

iv). Change of the path orientation

Concerning the path orientation change, the trajectories of components *3*, *4* and *5* in the *O,x,y* world coordinate system are shown in Fig. 5.15. Therefore, in this situation, the sub-assembly (*3,4*) will be in an unstable state, which will need more fixtures in order to insure the stability. Note that collision force feedback leads to the turbulence of the path' curves, as shown in Fig. 5-15.

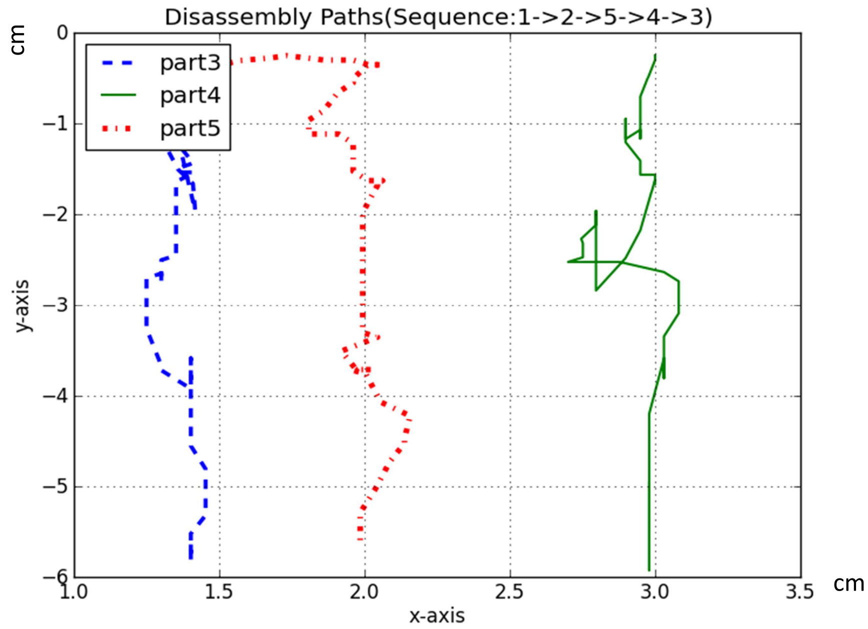
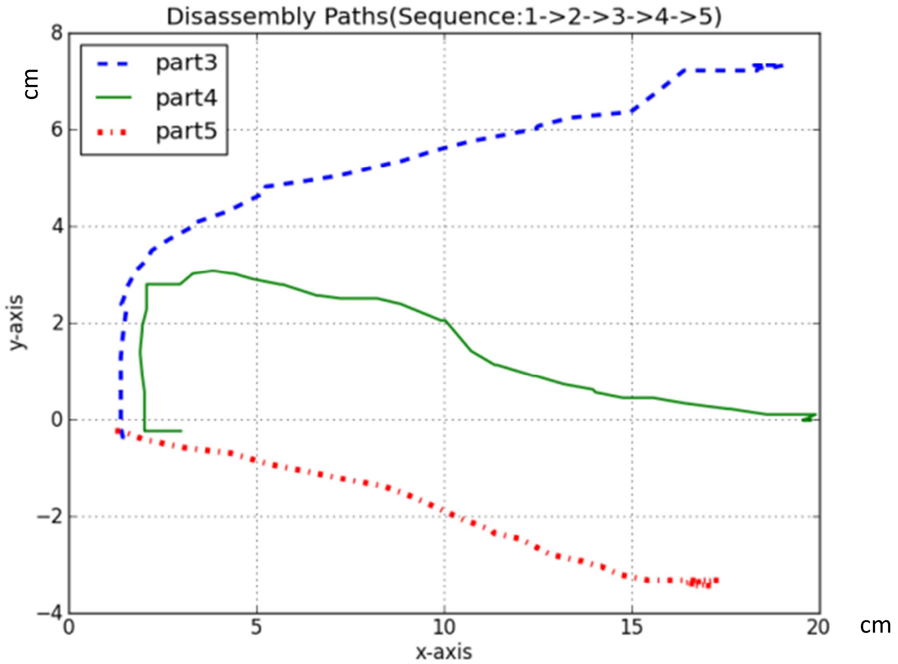


Figure 5.15 Trajectories for components 3 and 4 (unstable state) and the removing part 5 (causing this instability). (***x*** and ***y*** axes are the coordinates of the parts' center of gravity).

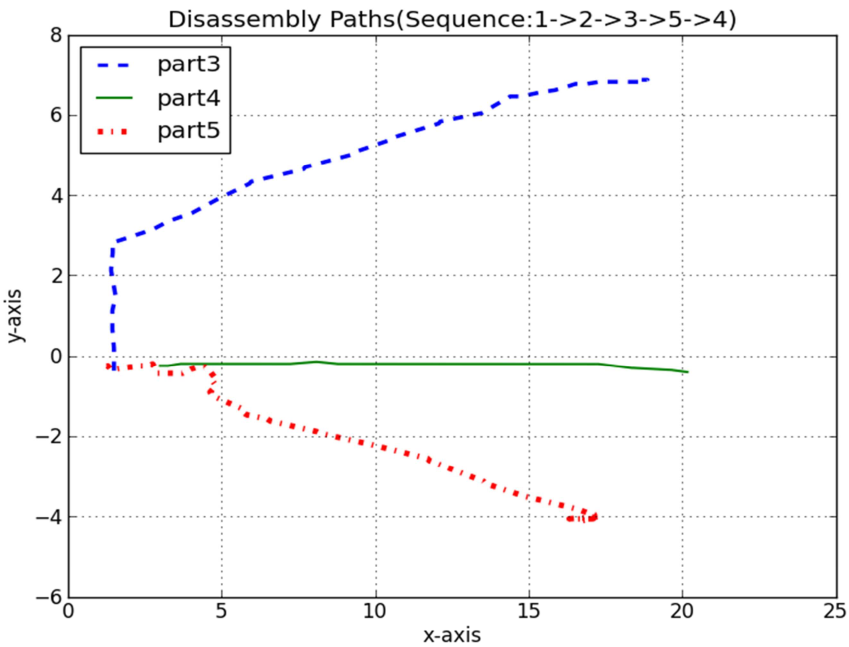
5.3.2 Results

In order to compare the trajectories of the different components, during the disassembly sequence, the *path* lines (trajectories) for parts *3*, *4* and *5* in *O,x,y* plane are recorded (Fig. 5.17). There are four possible disassembly sequences for this assembly, namely: {*1,2,3,4,5*}, {*1,2,3,5,4*}, {*1,2,5,3,4*} and {*1,2,5,4,3*}. It is noted that parts *1* and *2* have the same order in all these sequences. Their trajectories are the same and consequently it is useless to compare them.

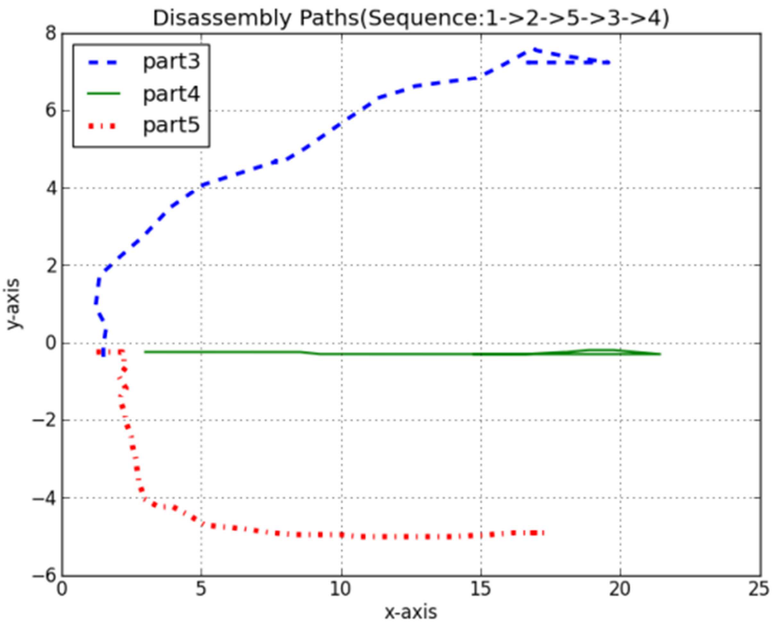
The paths orientation change being one of the criteria to evaluate the disassembly operation, the best one, for part *4*, belongs to sequences {*1,2,5,4,3*) (Fig.5.16d) as its path is nearly straight horizontal line. It may be pointed out that for sequences {*1,2,3,5,4*} (Fig.5.16b) and {*1,2,5,3,4*} (Fig.5.16c) the paths orientation change is also almost in straight lines. However, the worst path change, for part *4*, belongs to sequences {*1,2,3,4,5*} (Fig.5.16a), because it requires some steering to reach the destination surface.


Disassembly Paths(Sequence:1->2->3->4->5)
part3
part4
part5
cm
y-axis
x-axis
cm
8
6
4
2
0
−2
−4
0
5
10
15
20

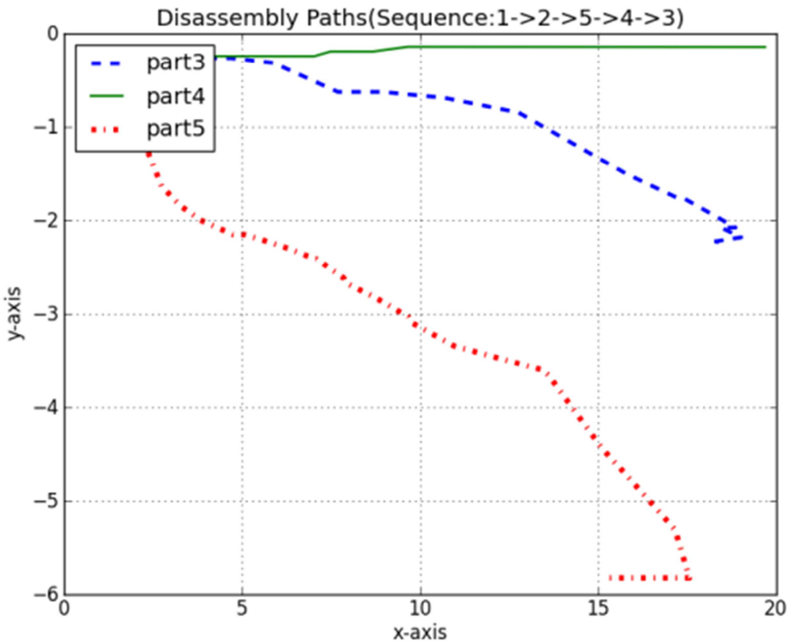
(a)


Disassembly Paths(Sequence:1->2->3->5->4)
part3
part4
part5
y-axis
x-axis
8
6
4
2
0
−2
−4
−6
0
5
10
15
20
25

(b)



(c)



(d)

Figure 5.16 Components' 3, 4 and 5 disassembly paths for the possible disassembly sequences (***x*** and ***y*** axes are the coordinates of the parts' center of gravity).

After performing the four disassembly sequences, the scores for the four proposed criteria are calculated (Table 5.2).

Table 5.2 Criteria scores for each sequence

	Ergonomic evaluation criteria			Traditional disassembly criteria				SUM
Sequences	Visibility of a part	Neck score (NS)	Bending score (BS)	Disass-embly angle	Number of tools' changes	Path orientation change	Sub-assembly stability	
(*1,2,5,3,4*)	*0.69857*	*0.55230*	*1.0*	*0.73333*	*0.0*	*0.52767*	*0.6*	*4.11187*
(*1,2,5,4,3*)	*0.68685*	*0.55230*	*1.0*	*0.68499*	*0.0625*	*0.52670*	*0.6*	*4.11334*
(*1,2,3,5,4*)	*0.66321*	*0.41563*	*1.0*	*0.71467*	*0.1667*	*0.77778*	*0.8*	*4.53799*
(*1,2,3,4,5*)	*0.60232*	*0.32414*	*0.79550*	*0.63333*	*0.1667*	*0.52123*	*1*	*4,04322*

The latest column presents the sum of the seven criteria scores for each disassembly sequence. The higher the value is the better sequence is. Thus, the best one is for sequence {1,2,3,5,4} with SUM=4.53799.

Let us note that, the values for *visibility of a part* and *path changing* depend on the way that the operator is handling the components in the virtual environment. However, the values of *disassembly angles (SDR)*, the *number of tools' changes* and the *stability* are not related to the operator's abilities and consequently only depend on the mechanical assembly and the disassembly sequence itself. Two subjects were involved in the performed experiment for disassembly simulations. In order to improve the reliability of the proposed method, the average duration of the disassembly time for these two subjects were recorded as well. The results for average disassembly time for each sequence are shown in Table 5.*3*.

Table 5.3 Duration time for each sequence

Sequences	Subject 1 time (seconds)	Subject 2 time (seconds)	Average time (seconds)
(1,2,5,3,4)	*48.4850001335*	*46.2220001221*	*47.3535001278*
(1,2,5,4,3)	*46.2139999866*	*44.1860001087*	*45.2000000476*
(1,2,3,5,4)	*40.4040000439*	*39.0890002011*	*39,7465001225*
(1,2,3,4,5)	*44.3980000019*	*44.6011113981*	*44,4995557000*

The shortest time is 39.0890002011 sec for sequence {1,2,3,5,4} performed by Subject 2, which is consistent with the previous evaluation thus showing that this sequence is the best evaluated one according to the proposed criteria.

5.4 Conclusion

This Chapter introduced a new method for evaluation of disassembly operations in Virtual Environment which combine ergonomic and traditional processing procedure evaluations.

Ergonomic evaluation involved detecting the distance and angle between the camera and the target components in order to evaluate the ergonomic parameters instead of using a human *DMU* model. For this purpose three criteria namely: *visibility score, neck score* and *bending score* were proposed. Thus, the overall score of the proposed three criteria gives enough information about the operation efficiency evaluation from an ergonomic viewpoint.

Concerning the traditional processing evaluation, it included a set of four criteria namely: *disassembly angles, stability of the subassembly, number of tools' changes* and *path direction change*. It allows evaluating the disassembly operation complexity during the initial stage of product design or during the Product Life Cycle (PLC) in: production process, product maintenance and at the end of PLC. The performed tests, whose a case study was presented here, demonstrated the efficacy of the proposed method. The score results of the seven criteria, divided in two categories, allowed selecting the best disassembly sequence. It was confirmed by experimental test thus allowing validating the proposed method. The method is validated by developing an application for virtual simulation based on *Python* programming language associated with *VTK* and *ODE* libraries. The application was tested by performing disassembly operation simulations evaluation and two examples in the case of five components disassembly and two screws from a mechanical assembly were presented. Thus, the method can be naturally adapted to a variety of virtual environment applications for A/D sequences evaluation. For the traditional processing evaluation the test results showed that the values of three criteria namely: *disassembly angles (SDR), number of tools' changes* and *stability* of the sub-assembly only depend on the complexity level of the mechanical assembly and consequently are not related to the operator's abilities. On the contrary, the values of: *visibilities of a part* and *path change* strongly depend on the way that the operator is performing the handling of the components in the VE.

General conclusions

Some limitations of the available techniques for disassembly operation simulations stimulated this thesis research on disassembly operations simulation including: *sequences generation* and *evaluation*. Different existing research methods in the field of disassembly sequencing were presented. We have pointed out that Automatic disassembly sequencing is an ideal way for the disassembly sequencing. However, there are two major problems in this field. One is the representing models for the product. There are many graph based methods or networks as presented above for representing the relationships among the parts in a product. However, the graph-based techniques for example, do not consider products geometrical information data bases. As we are aware, there are not works mentioning that these graphs or networks can be built automatically according to parts relationship in the product. The other problem is related with the calculation method. Basically, all the graphs can be translated into matrix calculation for sequences searching or calculation. Many works, based on some simplifications hypotheses, only focus on four or six directions to disassembly the product which is easy to transfer the disassembly calculation into the matrix computation model. However, in the real situation, the components disassembly direction cannot be just in four or six directions and the rotations have to be taken into account as well. After getting all the sequences of disassembly, the evaluation will be important for choosing the most efficiency of them. Thus, the aim of our work was to contribute to modeling of assembly/disassembly operations: sequences generation and their evaluation in a Virtual reality environment.

1. First, we have presented a method for disassembly sequences generation. The method consists in setting up a new approach for the sequences' generation called "*lowest level of disassembly graph method*", which is based on the notion of *disassembly geometry contacting graph* (DGCG). The graph is built on *collision* and *set of directions for removal (*SDR) detection for each given component in an assembly. For this purpose three cases, called *micro units*, which consider all the possible situations of relationships, among the components in the DGCG were addressed. With the investigated cases, the proposed method eliminates all the components which are unrelated to the target. The disassembly order graph is like a problem of inverted tree containing a minimum set of components related with the target component disassembly. Thus, the unrelated components are eliminated which allows to reduce the number of iterations for disassembly sequence generation and consequently search time. Compared with other existing methods which can be used in some special situations, for

example, the direction of disassembly must be certain, or the products must be relatively simple, our method can generate disassembly sequences for any kind of complexity of products as the process for DGCG generating automatically stops when the target component(s) is (are) reached. The efficiency of the proposed method was proved by its application, for disassembly sequences generation of different mechanical and electromechanical assemblies.

2. Secondly, our thesis proposed a new method for the evaluation of disassembly operations. For this purpose two sets of criteria have been proposed. The first one considers the traditional processing evaluation and consists of four criteria: disassembly angles, stability of the subassembly, number of tools' changes and path direction change. Then, the overall score, defined as the sum of them is automatically calculated by the realized application. It gives enough information about the operation efficiency evaluation from a technological (processing) viewpoint and allows evaluating the disassembly operation complexity by considering the real disassembly process. The second set of criteria concerns the ergonomic evaluation and consists of*: visibility score*, *neck score* and *bending score*. The purpose of the Ergonomical assessment being trying to fit the task to the human and not the human to the task, this evaluation is focusing on the convenient of human body while performing disassembly operations. Based on the proposed methods for *disassembly sequences generation* and *disassembly operation evaluation* an application for virtual simulation based on Python programming language associated with VTK and ODE libraries was developed.

The key point for an effective application is to gain a balance between the human body features and the task demands. Instead of the ergonomics simulation with a human model, the realized application introduces some new sources in performing disassembling task in a VR environment.

Thus, the score result of the proposed two sets of criteria allowed to select the best disassembly sequence. It was confirmed by experimental tests thus allowing validating the proposed method. Consequently, it can be naturally used to a variety of virtual environment applications for A/D sequences evaluation. The proposed application can be implemented into any existing industrial software, or design tools in particular for product disassembly simulation. The analysis results and findings demonstrate the feasibility of the proposed approaches, thus providing significant assistance for the evaluation of disassembly sequences during Product Development Process.

Assessment and Prospects

Modelling disassembly operations requires a lot of geometrical, kinematical, technological and ergonomical data and their synthesis in order to reduce the algorithmic complexity of the disassembly simulation process. Nowadays, disassembly operation simulation of industrial products finds a strong interest in interactive simulations through immersive and real-time schemes.

The majority of the works related with disassembly sequences generation and disassembly operation evaluation often require tremendous computational resources while, they often fail to find realistic and optimal solutions for complex products disassembly. Virtual assembly simulations allow the designer to evaluate the concepts in virtual environments during the early design stage. With virtual prototyping applications, optimizing design process for the design for assembly can be incorporated easily in the conceptual design stage. Using haptics or auditory technology, allows designers to interact with the parts with the human basic motions.

As known, the number of possible disassembly sequences increases significantly with the number of parts in a product. Thus, the generation of proper disassembly sequences order is critical.

In the proposed method for disassembly sequence generation the created *DGCG* contains a minimum set of components related to the target. As previously said the unrelated components are eliminated in order to reduce the number of iterations and search time.

At this stage, the proposed VR environment is not completed because the presented work is limited only for disassembly sequences generation and their evaluation. However, it is the base for further extensions and realizations. In a near future it could be the object of a continuation of studies.

For improvements, we can extend the study in the following direction:

- 1. Concerning the method for disassembly sequences generation in virtual reality environment; future work will consider the integration of the proposed method for disassembly process sequence generation and evaluation in a virtual reality (VR) system with perception model. Considering the lowest level of disassembly to generate the possible sequences, how to choose the best one with lowest cost value in the real disassembly process is still an issue.

- 2. Concerning the method for evaluation of disassembly sequences the score sums of the proposed two sets of criteria give enough information about the sequence's efficacy evaluation. However at this stage our method does not consider the ranking of the criteria thus proposed, as they have the same weight. Thus, future work consists in ranking the criteria according to their importance. For this purpose moderation coefficients can be allocated to each of them thus allowing a more comprehensive evaluating method.

References

[Ale 11] Aleotti J., Caselli S., *Physics-based virtual reality for task learning and intelligent disassembly planning*. Virtual reality. 15:41-54, 2011.

[Ash 09] Ashvinikumar P., Dibakar, S., *Haptics Aided kinematic Assembly Modeling and Efficient Determination of Joint Ranges of Motion*. 14-th National conference on Machines and Mechanisms, NIT, Durgapur, India, December 17-18, pp. 68-75, 2009.

[Ats 13] Atsuko E., Noriaki Y., Tasuya S., *Automatic estimation of the ergonomics parameters of assembly operation*. CIRP Annals-Manufacturing Technology. 62(1):13-16, 2013.

[Bal 91] Baldwin D.F., Abell T.E., Lui M.C., de Fazio T.L, Whitney D.E., *An integrated computer aid for generating and evaluating assembly sequences for mechanical products*. IEEE Trans Robot Automat 7(1):78–89, 1991.

[Bar 04] Barnes C.J., Jared G.E.M., Swift K.G., *Decision support for sequence generation in an assembly oriented design environment*, Robotics and Computer Integrated Manufacturing – RCIM, vol. 20, p. 289-300, Ed. Elsevier, 2004.

[Bou 84] Bourjault A., *Contribution à une approche méthodologique de l'assemblage automatise: élaboration automatique des séquences opératoires*. Dissertation, d'Etat Université de Franche-Comté, Besançon, France, 1984.

[Bou 02] Bouzit M., Popescu G., Burdea G.C., Boian R., *The Rutgers master II-ND force feedback glove*. In HAPTICS 2002: haptic interfaces for virtual environment and teleoperator systems. Orlando, FL, 2002.

[Cam 90] Cameron S., *Collision detection by four-dimensional intersection testing*. IEEE Transactions on Robotics and Automation, *6* (3): 291-302, 1990.

[Che 01] Chen S.F., Liu Y.J., *The application of multilevel genetic algorithms in assembly planning,* Journal of Industrial Technology, vol. 17, n°. 4, Ed. ATMAE, 2001.

[Cru 92] Cruz-Neira C., Sandin D.J., DeFanti T.A., Kenyon R., Hart J.C., *The CAVE, audiovisual experience automatic virtual environment*. In: Communications of the ACM, pp 64-72, 1992.

[Cru 93] Cruz-Neira C., Sandin D., De Fanti T., *Surround-screen projection-based virtual reality: the design and implementation of the CAVE*. In: Proceeding of SIGGRAPH, l93: 135-142, 1993.

[Don 06] Dong T., Zhang L., Tong R., Dong J., *A hierarchical approach to disassembly* *sequence planning for mechanical product*, International Journal of Advanced Manufacturing Technology, 30 (5–6):507–520, 2006.

[Duv 13] Duval T., Huyen Nguyen T.T., Fleury C., Chauffaut A., Dumont G., Gouranton V., *Improving awareness for 3D virtual collaboration by embedding the features of users' physical environments and by augmenting interaction tools with cognitive feedback cues*. In Journal on Multimodal Interfaces (JMUI), 2013. doi: 10.1007/s12193-013-0134-z.

[Edw 04] Edwards G.W., Barfield W., Nussbaum M.A., *The use of force feedback and auditory cues for performance of an assembly task in an immersive virtual environment.* Virtual Reality, 7:112-119, 2004.

[Erl 05] Erleben K., Sporrting J., Henriksen K., Dohlmann H., *Physics-based animation*, Carles River Media, Hingham, pp817, 2005.

[Faz 87] De Fazio T.L., Whitney D.E., *Simplified Generation of All Mechanical Assembly Sequences*, IEEE Journal of Robotics and Automation, 3 (6) : 640–658, 1987.

[Foi 93] Foisy A., Hayward V., *A safe swept volume method for collision detection*. The Sixth International Symposium of Robotics Research, Pittsburgh, PE, October, pp.61-8, 1993.

[Fuc 06] Fuchs P., *Le traité de la réalité virtuelle, vol. 1: L'homme et l'environnement virtuel*, Presses de l'Ecole des Mines, Paris, 2006.

[Gal 04] Galantucci L.M., Percoco G., Spina R., *Assembly and disassembly by using fuzzy logic and genetic algorithms,* International Journal of Advanced Robotics System, 1 (2): 67–74, 2004.

[Gao 98] Gao X.S., Chou S.C., *Solving geometric constrain systems. II. A symbolic approach and decision of re-constructability*. Commuter-aided design 30(2):115-112, 1998.

[Gar 04] Garcia M.A., Larré A., Lopez B., Oller A., *Reducing the complexity of Geometric Selective Disassembly*, In: Proceeding of the IEEE international Conference on Intelligent Robots and Systems, p.1474-1479, Takamatsu, Japan, 2004.

[Gar 07] Garbaya S., Zaldivar-Colado, U., *The effect of contact force sensations on user performance in virtual assembly tasks.* Virtual Reality, 11(4):287–299, 2007.

[Gei 96] Geigera D., Zussmannb E., Lenz E., *Probabilistic Reactive Disassembly Planning*. CIRP Annals - Manufacturing Technology. 45(1): 49–52, 1996.

[Ger 13] Germanico G-B., Medellin-Castillo H., Lim T., *Development of a Haptic Virtual Reality System for Assembly Planning and Evaluation Procedia Technology*, 7: pp. 265-272, 2013.

[Giu 07] Giudice F., Fargione G., *Disassembly planning of mechanical systems for service and recovery: A genetic algorithms based approach.* Journal of Intelligent Manufacturing, 18: 313-329, 2007.

[Go 12] Go T.F., Wahab D.A., Ab. Rahman M.N., Ramli R., Hussain A., *Genetically optimized disassembly sequence for automotive component reuse.* Expert Systems with Application 39:5409-5417, 2012.

[Got 03] Gottipolu R.B., Ghosh K., *A simplified and efficient representation for evaluation and selection of assembly sequences*, Computers in Industry 50(3):251-264, 2003.

[Gun 01] Gungor A., Gupta S. M., *Disassembly sequence plan generation using a branch-and-bound algorithm.* International Journal of Production Research, 39(3):481-509, 2001.

[Hom 90] Homem De mello L.S, Sanderson A.C., *AND /OR graph representation of assembly plan.* IEEE Transactions on Robotics and Automation 6(2):188-99, 1990.

[Hom 91] Homem de Mello L.S, Sanderson A.C., *A correct and complete algorithm for the generation of mechanical assembly sequences*. IEEE Trans Robot Automat 7(2): 228–240, 1991.

[Hsi 08] Hsieh F.S., *Robustness analysis of holonic assembly/disassembly processes with Petri nets*. Automatuca 44:2538-2548, 2008.

[Hua 00] Huang H.H., Wang M.H, Johnson M.R., *Disassembly sequence generation using a neural network approach*. Journal of Manufacturing Systems Volume 19(2), pp. 73–82. 2000.

[Iac 08] Iacob R., Mitrouchev P., Léon J-C., *Contact identification for assembly/disassembly simulation with a haptic device.* The Visual Computer, ISSN: 0178-2789, Ed. Springer, 24 (11), 973-979, 2008.

[Iac 10] Iacob R., *Modélisation cinématique des mobilités de composants pour des opérations d'assemblage et de désassemblage*, Thèse de Doctorat, Grenoble INP, Octobre 2010.

[Iac 14] Iacob R., Popescu D., Noel F., Louis T., Mitrouchev P., Larcher A., Assembly simulation using haptic devices, MIT 2014 Conference Proceedings, Fiesa, Slovenia, 27.09-01.10 2014.

[Jay 99] Jayaram S., Jayaram U., Wang Y., Tirumali H., Lyons K., Hart P., *VADE: A virtual assembly design environment*, IEEE Comput Graph Appl, 19(6):44-50, 1999.

[Jay 06] Jayaram U., Jayaram S., Shaikh I., Kim Y., Palmer C., *Introducing quantitative analysis methods into virtual environment for real-time and continuous ergonomic evaluations.* Computers in Industry 57:283-296, 2006.

[Jay 07] Jayaram S., Jayaram U., Kim A., De Chenne C., Lyons K., Palmer C., Mitsui T., *Industry case studies in the use of immersive virtual assembly*. Virtual reality 11(4):217-218, 2007.

[Jim 01] Jiménez P., Thomas F., Torras C., *3D collision detection: a survey*, Computers and Graphics, 25:269-285, 2001.

[Joh 98] Johnson M.R., Wang M.H., *The economical evaluation of disassembly operations for recycling, remanufacturing and reuse*, Int. Jour. Prod. Research, 36 (12), 3227-3252, 1998.

[Jun 03] Jung B., Task-level Assembly Modeling in Virtual Environments. Computational Science and its Applications, 2669:721-730, 2003.

[Kan 01] Kang J.G., Lee D.H., Xirouchakis P., Persson J.G., *Parallel disassembly sequencing with sequence-dependent operation times*. Annals of CIRP 50(1):343-6, 2001.

[Kar 06] Kara S., Pornprasitpol P., Kaebernick H., *Selective Disassembly Sequencing: A Methodology for the Disassembly of End-of-Life Products*, CIRP Annals, 55 (1):37–40, 2006.

[Kim 03] Kim C.E., Vance J.M., *Using Vps (Voxmap Pointshell) as the basis for interaction in a virtual assembly environment*. In: ASME design engineering technical conferences and computers and information in engineering conference (DETC2003/CIE-48297). ASME, Chicago, IL, 2003.

[Kok 07] Kok A.J.F, Van Liere R., A Multimodal Virtual Reality Interface for 3D Interaction with VTK,Knowledge and Information Systems, Knowledge and Information Systems, 2007

[Kon 06a] Kongar E., Gupta S.M., *Disassembly sequencing using genetic algorithm*. Int. Jour. Adv. Manuf. Technol., 30, pp. 497–506, 2006.

[Kon 06b] Kongar E., Gupta, S.M., *Genetic algorithm for disassembly process planning*. In Proceedings of the SPIE international conference on environmentally conscious manufacturing (Vol. II) (pp.52-62). Newton: Massachusetts, 2006.

[Kuo 00a] Kuo T.C., *Disassembly sequence and cost analysis for electromechanical products*, Robotics and Computer Integrated Manufacturing, 16:43-54, 2000.

[Kuo 00b] Kuo T.C., Zhang H.C., Huang S.H., *Disassembly analysis for electromechanical products: a graph-based heuristic approach*. Int. J. Prod. Res., 38 (5), pp. 993- 1007, 2000.

[Kuo 01] Kuo T.C., Huang, S.H., Zhang, H.C., *Design for manufacture and design for 'X': concepts, applications, and perspectives*, Computers & Industrial Engineering N°41, pp. 241-260, 2001.

[Kuo 13] Kuo T.C., *Waste electronics and electrical equipment disassembly and recycling using petri net analysis*: Considering the economic value and environmental impacts. Computers &Industrial Engineering 65, 54-64, 2013.

[Lad 10] Ladeveze N., Fourquet J.Y, Puel B., *Interactive path planning for haptic assistance in assembly tasks.* Computers &Graphics 34: 12-25, 2010.

[Lam 03] Lambert A.J.D., *Disassembly sequencing: a survey*, Int. Jour. of Prod. Res., 41(16):3721-3759, 2003.

[Lam 08] Lambert A.J.D., Gupta M., *Methods for optimum and near optimum disassembly sequencing*, Int. Jour. of Prod. Res., 46, 11, 2845-2865, (2008).

[Leu 13] Leu M.C., ElMaraghy, H.A., Nee, A.Y.C., Ong, S.K., Lanzetta M., Putz M., Zhu W., Bernard A., *CAD model based virtual assembly simulation, planning and training.* CIRP Annals-Manufacturing Technology, 62(2):799-822, 2013.

[Lee 94] Lee S., *Subassembly Identification and Evaluation for Assembly Planning*. IEEE Transactions on Systems Manufacturing, and Cybernetics, 24 (3):493–503, 1994.

[Li 12] Li J.R., Wang, Q.H. Huang P., *An integrated disassembly constraint generation approach for product design evaluation.* International Journal of Computer Integrated Manufacturing, 25(7):565-577, 2012.

[Lon 06] Longo F., Mirabelli G., Papoff. E., *Effective Design of Assembly Line Using Modelling & Simulation*. Proceedings of the 2006 Winter Simulation Conference, pp. 1894–1898, 2006.

[Mar 03] Luis M., Norman M., Terrence F. *A constraint manager to support virtual maintainability.* Computers & Graphics 27(1):19-26, 2003.

[Mas 03] Mascle C., Balasoiu B.A., *Algorithmic selection of a disassembly sequence of a component by a wave propagation method*, Robotics and Computer-Integrated Manufacturing, Vol. 19, Issue 5, October, pp. 439–448, 2003.

[McA 93] McAtamney L., Corlett E.N., *RULA: a survey method for the investigation of work-related upper limb disorders*, Applied Ergonomics, 24(2):91-99, 1993.

[Moo 98] Moore K.E., Askiner G., Surendra M. Gupta. A., *Petri Net Approach to disassembly process planning.* Computer Ind. Engng., 35(1-2):165-168, 1998.

[Moo 01] Moore K.E., Gungor A., Surendra M. Gupta A., *Petri net approach to disassembly process planning for products with complex AND/OR precedence relationships*. European Journal of Operational Research 135:428-449, 2001.

[Mo 02] Mo J., Zhang O., Gadh R., Virtual Disassembly, International Journal of CAD/CAM, 2(1), 29-37, 2002.

[Ou 13] Ou L.M, Xu X., *Relationship matrix based automatic assembly sequence generation from a CAD model*, Computer-Aided Design 45:1053-1067, 2013.

[Per 13] Perret F., Kneschke C., Vance F., Dumont G., *Interactive assembly simulation with haptic feedback.* Assembly Automation, Vol. 33 Iss: 3, pp.214 – 220, 2013.

[Pom 04] Pomares J., Puente S.T., Torres F., Candelas F.A., Gil P., *Virtual disassembly of products based on geometric models*. Computers in Industry, 55(1):1-14, 2004.

[Pon 13a] Pontonnier C., Samani A., Badawi M., Madeleine P., Dumont G., *Assessing the Ability of a VR-based Assembly Task Simulation to Evaluate Physical Risk Factors*, In: IEEE Transactions on Visualization and Computer Graphics, Vol. 20, No. 5, 2014, pp. 664-674.

[Pon 13b] Pontonnier C., Dumont G., Samani A., Madeleine P., Badawi M., *Designing and Evaluating a Workstation in Real and Virtual Environment: Toward Virtual Reality Based Ergonomic Design Sessions,* Journal on Multimodal Interfaces http://www.springer.com/computer/hci/journal/12193 (JMUI), 2013. http://link.springer.com/article/10.1007/s12193-013-0138-8.

[Pon 13c] Pontonnier C., Duval T., Dumont G., *Sharing and Bridging Information in a Collaborative Virtual Environment: Application to Ergonomics*, Proceedings of 4-th IEEE International Conference on Cognitive Infocommunications (CogInfoCom 2013), Budapest, Hungary, December 2013, pp.121-126., 10.1109/CogInfoCom. 2013. 6719226.

[Pop 00] Popescu V.G., Burdea G.C., Bouwit M., Hentz V.R., *A virtual–reality–based telerehabilitation system with force feedback.* IEEE Inf. Technol. Biomed., 4(1):45-51, 2000.

[Ren 05] Renouf M., Acary V., Dumont G., *Comparison of Algorithms for collisions, contact and friction in view of Real-time applications*. Multibody Dynamics 2005 : International Conference on Advances in Computational Multibody Dynamics, ECCOMAS Thematic Conference, 21-24 June 2005.

[Ric 95] Richard P., Coiffet P. *Human perceptual issues in virtual environments: sensory substitution and information redundancy*, Robot and Human Communication, 1995. RO-MAN'95 TOKYO, Proceedings 4-th IEEE International Workshop on Robot and Human Communication, pp.301-306, 1995.

[Ric 13] Rickli J., Camelio A., *Multi-objective partial disassembly optimization based on sequence feasibility*. Journal of Manufacturing Systems, 32(1), pp. 281–293, 2013.

[Sal 97] Salisbury J.K., Srinivasan M.A., *Projects in VR: phantom based haptic interaction* *with virtual objects.* IEEE Comp. Graph. Appl., Sept./Oct, 1997.

[San 02] Santochi M., Dini G., Failli F., *Computer Aider Disassembly Planning: State of the Art and Perspectives, CIRP Annals-Manufacturing Technology*, 51(2):507-529, 2002.

[Sch 96] Schroeder W. J., Martin K. M. and. Lorensen W. E., *The design and implementation of an object-oriented toolkit for 3D graphics and visualization*. IEEE Visualization '96, pages 93–100, 1996.

[Set 11] Seth A., Vance J.M., James H.O., *Virtual reality for assembly methods prototyping: a review*. Virtual Reality 15: 5-20, 2011.

[Sid 97] Siddique Z., Rosen D.W., *A virtual prototyping approach to product disassembly reasoning*, Computer-Aided Design, Vol. 29, p.847-860, 1997.

[Smi 11] Smith S.S., Chen W.H., *Rule-based recursive selective disassembly sequence planning for green design*, Advanced Engineering Informatics, vol. 25, p. 77-87, 2011.

[Smi 12] Smith S., Smith G., Chen W.H., *Disassembly sequence structure graphs: An optimal approach for multiple-target selective disassembly sequence planning*, Adv. Eng. Informat, 26 (2) 306-316, 2012.

[Sri 98] Srinivasan H., Gadh R., *A geometric algorithm for single selective disassembly using wave propagation abstraction*. Computer-Aided Design, Vol. 30, Issue 8, pp. 603-613, 1998.

[Sri 99a] Srinivasan H., Gadh R., *Selective disassembly: representation and comparative analysis of wave propagation abstractions in sequence planning*, Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning, Porto, Portugal, pp.129-133, 1999.

[Sri 99b] Srinivasan H., Figueroa R., Gadh R., *Selective disassembly for virtual prototyping as applied to de-manufacturing*, Robotics and Computer Integrated Manufacturing 15 (3) :231-245,1999.

[Sri 00] Srinivasan H., Gadh R., *Efficient geometric disassembly of multiple components from an assembly using wave propagation*, Journal of Mechanical Design, 122 (2):179-184, 2000.

[Su 07] Su Q., *Computer aided geometric feasible assembly sequence planning and optimizing*, Int. Jour. Adv. Manuf. Technol., 33: 48-57, 2007.

[Tar 13] Taroni F., Vitae F., Biedermann A., *Bayesian Networks Encyclopedia of Forensic Sciences* (Second Edition), pp. 351–356, 2013.

[Tch 10] Tching L., Dumont G. Perret J. Interactive simulation of CAD models assemblies using virtual constraint guidance, Int. Jour. Des. Manuf. (2010), 4:95-102.

[Tri 09] Tripathi M., Agrawal S., Pandeyc M.K., Shankar R., Tiwari M.K., *Real world disassembly modeling and sequencing problem: Optimization by Algorithm of Self-Guided Ants* (ASGA), Robotics and Computer-Integrated Manufacturing 25,483–496, 2009.

[Tse 09] Tseng Y., Kaoa H., Huang F., *Integrated assembly and disassembly sequence planning using a GA approach*. International Journal of Production Research, pp.1-23, 2009.

[Vee 02] Veerakamolmal P. Gupta S.M., *A case-based reasoning approach for automating disassembly process planning*. Journal of Intelligent Manufacturing 13(1): 47-60, 2002.

[Wan 01] Wang Y, Jayaram S., Jayaram U., Lyons K., *Physically based modeling in virtual assembly*. In: ASME design engineering technical conferences and computers and information in engineering conference (DETC2001/CIE-21259). Pittsburg, PA, 2001.

[Wan 12] Wang D., Zhang L., Wang M., Xiao T., Hou Z., Zou F., *A simulation System Based on OGRE and PhysX for Flexibal Aircraft Assembly*. IEEE/SCS 26-th Workshop on Principles of Advanced and Distributed Simulation, pp. 171-173, 2012.

[Wan 03] Wang J.F., Liu J.H., Li S.Q., Zhong Y.F., *Intelligent selective disassembly using the ant colony algorithm.* Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM, vol. 17, no. 4-5, pp. 325-333, 2003.

[Wan 06] Wang Q.H., Li J.R., Gong H.Q. *A CAD-linked virtual assembly environment*. Int. Jour. Prod. Res., 44(3):467-486, 2006.

[Wil 94] Wilson R., Latombe J., *Geometric reasoning about assembly*, Artif. Intell. 71 (2):371-396, 1994.

[Wil 99] Wilson J.R., *Virtual environments applications and applied ergonomics*, Applied Ergonomics, 30(1): 3-9, 1999.

[Woo 91] Woo T.C., Dutta D., *Automatic disassembly and total ordering in three dimensions*. Journal of Engineering for Industry, 113 (2), 207-213, 1991.

[Woo 94] Woo T.C., *Visibility maps and spherical algorithms*, Computer-Aided Design Journal. 26 (1), 6-16, 1994.

[Yan 07] Yang R., Wu D., Fax X., Yan J., *Research on constraint-based virtual assembly technologies.* Front. Mech. Eng. China 2(2):243-249, 2007.

[Yi 07] Yi J., Yu B., Du L., Li C., Hu D., *Research on the selectable disassembly strategy of mechanical parts based on the generalized CAD model*. The International Journal of

Advanced Manufacturing Technology, 37:599–604, 2007.

[Zei 97] Zeid I., Gupta S.M., Bardasz T., *A case-based reasoning approach to planning for disassembly*. Journal of Intelligent Manufacturing 8, pp. 97–106, 1997.

[Zha 10] Zhang X., Zhang S. Y., *Product cooperative disassembly sequence planning based on branch-and-bound algorithm*. Int Jour. Adv. Manuf. Technol., 19(4): pp. 91–103, 2010.

[Zhu 13] Zhu B.C, Sarigecili M.I, Roy U., *Disassembly information model incorporating dynamic capabilities for disassembly sequence generation*. Robotics and Computer-Integrated Manufacturing, 29(5):396-409, 2013.

[Zus 95] Zussman E., Scholz-Reiter G., Sharke H., *Modelling and planning of disassembly processes*. In: Life-Cycle Modelling for Innovative Products Processes. Berlin, Germany, pp. 221-232, 1995.

Appendix A

Part of code of Interface, Virtual Reality Disassembly Environment (VRDE)

Code in Python

```
import vtk
import wx,os,sys
import wx.lib.buttons as buttons
from vtkactors import *

from pylab import*
import areaReading
import matplotlib.pyplot as plt
import wx.lib.imagebrowser as ib

def vtk_Camera_Arrow():
   #creat the arrow for the pic
   arrowSource = vtk.vtkArrowSource()
   mapper = vtk.vtkPolyDataMapper()
   translation=vtk.vtkTransform()
   translation.RotateZ(180)
   translation.Translate(0.8,0,0)
   translation.Scale(2.0,2.0,2.0)
   transformFilter =vtk.vtkTransformPolyDataFilter()
   transformFilter.SetInputConnection(arrowSource.GetOutputPort())
   transformFilter.SetTransform(translation)
   mapper.SetInputConnection(transformFilter.GetOutputPort())
   actor = vtk.vtkActor()
   actor.SetMapper(mapper)
   actor.GetProperty().SetColor(1.0,0.0,0.0)
   #creat the camera for the simulation
   camCS = vtk.vtkConeSource()
   camCS.SetHeight(1.5)
   camCS.SetResolution(12)
   camCS.SetRadius(0.4)
   camCBS = vtk.vtkCubeSource()
   camCBS.SetXLength(1.5)
   camCBS.SetZLength(0.8)
   camCBS.SetCenter(0.4, 0, 0)
   camAPD = vtk.vtkAppendFilter()
   camAPD.AddInput(camCS.GetOutput())
   camAPD.AddInput(camCBS.GetOutput())
   camMapper = vtk.vtkDataSetMapper()
   camMapper.SetInput(camAPD.GetOutput())
   camActor =vtk.vtkLODActor()
   camActor.SetMapper(camMapper)
   #creat assembly
   assembly=vtk.vtkAssembly()
```

```
    assembly.AddPart(camActor)
    assembly.AddPart(actor)
    assembly.SetScale(20,20,20)
    return assembly
def position2matrix(pos, scale) :
    # pos is a vector x,y,z,qx,qy,qz,qw
    x = pos[0]*scale
    y = pos[1]*scale
    z = pos[2]*scale
    qx = pos[3]
    qy = pos[4]
    qz = pos[5]
    qw = pos[6]
    m0= 1 - 2 * ( (qy*qy) + (qz*qz) )
    m1=    2 * ( (qx*qy) - (qz*qw) )
    m2=    2 * ( (qx*qz) + (qy*qw) )
    m4=    2 * ( (qx*qy) + (qz*qw) )
    m5= 1 - 2 * ( (qx*qx) + (qz*qz) )
    m6=    2 * ( (qy*qz) - (qx*qw) )
    m8=    2 * ( (qx*qz) - (qy*qw) )
    m9=    2 * ( (qy*qz) + (qx*qw) )
    m10= 1 - 2 * ( (qx*qx) + (qy*qy) )
    m3=x
    m7=y
    m11=z
    m12=m13=m14= 0
    m15= 1
    return ((m0,m1,m2,m3),
    (m4,m5,m6,m7),
    (m8,m9,m10,m11),
    (m12,m13,m14,m15))

def Keypress(obj, event):
    key = obj.GetKeySym()
    if key == "e":
        obj.InvokeEvent("DeleteAllObjects")
        sys.exit()
    elif key == "w":
        Wireframe()
    elif key =="s":
        Surface()

def vtk_Cube(x,y,z,r=0.5,g=0.5,b=0.5):
    cyl=vtk.vtkCubeSource()
    cyl.SetXLength(x)
    cyl.SetYLength(y)
    cyl.SetZLength(z)
    mapper =vtk.vtkPolyDataMapper ()
    mapper.SetInputConnection(cyl.GetOutputPort())
    lxactor =vtk.vtkActor()
    lxactor.SetMapper(mapper)
    lxactor.GetProperty().SetColor (r, g, b)
```

```
    lxactor.flag="cube"
    lxactor.SetScale(40,40,40)
    return lxactor

def vtk_cylinder(radius,h,resolution,r,g,b):
    cy=vtk.vtkCylinderSource()
    cy.SetRadius(radius)
    cy.SetHeight(h)
    cy.SetResolution(resolution)
    translation =vtk.vtkTransform()
    translation.RotateX (90.0)
    transformFilter =vtk.vtkTransformPolyDataFilter()
    transformFilter.SetInputConnection(cy.GetOutputPort())
    transformFilter.SetTransform(translation)
    mapper =vtk.vtkPolyDataMapper()
    mapper.SetInputConnection(transformFilter.GetOutputPort())
    lxactor =vtk.vtkActor()
    lxactor.SetMapper(mapper)
    lxactor.GetProperty().SetColor (r, g, b)
    return lxactor


def vtk_sphere(radius,theta,phi,r,g,b):
    sphereSource=vtk.vtkSphereSource()
    sphereSource.SetRadius(radius)
    sphereSource.SetThetaResolution (theta)
    sphereSource.SetPhiResolution (phi)
    actor1=vtk.vtkActor()
    mapper=vtk.vtkPolyDataMapper()
    mapper.SetInputConnection(sphereSource.GetOutputPort())
    actor1.SetMapper(mapper)
    actor1.GetProperty().SetColor (r,g,b)
    actor1.flag="sphere"
    return actor1
address=["C:\chenggang\ode\pics\Blue_Grey_Granite.bmp","C:\chenggang\ode\pics\Buff_Quartz.b
mp"]

def bmpReader(address):
    bmpReader = vtk.vtkBMPReader()
    bmpReader.SetFileName(address)
    C=vtk.vtkTexture()
    C.SetInput(bmpReader.GetOutput())
    C.InterpolateOn()
    return C

def creat_Plane():
    plane = vtk.vtkPlaneSource()
    plane.SetPoint1(300.0, 10.0, 300.0 )
    plane.SetPoint2(-300.0, 10.0, 300.0 )
    plane.SetCenter(0.0, 1.0, 0.0)
    plane.SetXResolution(100)
    plane.SetYResolution(100)
```

```
    planeMapper = vtk.vtkPolyDataMapper()
    planeMapper.SetInput(plane.GetOutput())
    planeActor = vtk.vtkActor()
    planeActor.SetMapper(planeMapper)
    planeActor.SetTexture(bmpReader(address[0]))
    planeActor.PickableOff()
    return planeActor

class myFrame(wx.Frame):##
    def __init__(self,parent,title):
        wx.Frame.__init__(self,parent,title=title,size=(550,300))##
        self.RenderWindow=vtk.vtkRenderWindow()
        self.RenderWindow.StereoCapableWindowOn()
        self.RenderWindow.SetSize(800, 800)
        self.Renderer=vtk.vtkRenderer()
        self.RenderWindow.AddRenderer(self.Renderer)
        self.Renderer.TwoSidedLightingOn ()
        self.Renderer.LightFollowCameraOn ()
        self.Renderer.SetBackground(0.1,0.3,0.5)
        self.Rotating = 0
        self.Panning = 0
        self.cameraRotating = 0
        planeActor=creat_Plane()
        #self.Renderer.AddActor(planeActor)
        self.actor = None
        self.iren=vtk.vtkRenderWindowInteractor()
        self.iren.SetInteractorStyle(None)
        self.iren.AddObserver("LeftButtonPressEvent", self.ButtonEvent)
        self.iren.AddObserver("LeftButtonReleaseEvent", self.ButtonEvent)
        self.iren.AddObserver("MiddleButtonPressEvent", self.ButtonEvent)
        self.iren.AddObserver("MiddleButtonReleaseEvent",self.ButtonEvent)
        self.iren.AddObserver("RightButtonPressEvent", self.ButtonEvent)
        self.iren.AddObserver("RightButtonReleaseEvent", self.ButtonEvent)
        self.iren.AddObserver("MouseWheelForwardEvent", self.ButtonEvent)
        self.iren.AddObserver("MouseWheelBackwardEvent", self.ButtonEvent)
        self.iren.AddObserver("MouseMoveEvent", self.MouseMove)
        self.iren.AddObserver("KeyPressEvent", self.Keypress)
        self.iren.SetRenderWindow(self.RenderWindow)

        self.transform=((1.,0,0,0),(0,1.,0,0),(0,0,1.,0),(0.,0.,0.,1.))
        self.row=0
        self.col=0
        self.nbcol=2
        self.mode=301
        self.sphereX=[]
        self.sphereY=[]
        self.t2=0
        self.CreateStatusBar()#
        filemenu=wx.Menu()#
        open=filemenu.Append(wx.ID_OPEN,"&Open","Open the program")
        save=filemenu.Append(wx.ID_SAVE,"&Save","Save the program")
        about=filemenu.Append(wx.ID_ABOUT, "&About","Information about this program")
```

```
    filemenu.AppendSeparator()# 不懂
    exit=filemenu.Append(wx.ID_EXIT,"&Exit","Terminate the program")

    self.Bind(wx.EVT_MENU,self.OnAbout,about)#选择 about 将执行 onAbout 函数
    self.Bind(wx.EVT_MENU,self.OnExit,exit)
    self.Bind(wx.EVT_MENU,self.OnOpen,open)
    self.Bind(wx.EVT_MENU,self.OnSave,save)

    ####菜单二
    ID_MENU_TRA = wx.NewId()
    ID_MENU_TIME = wx.NewId()
    ID_MENU_CollISION = wx.NewId()
    ID_MENU_TOOL = wx.NewId()
    ID_MENU_REACH = wx.NewId()
    filemenu2=wx.Menu()#菜单
    Tar=filemenu2.Append(ID_MENU_TRA,"&Path orientation changing ","All the Trajectories for all
the components")
    TIME =filemenu2.Append(ID_MENU_TIME, "&The stability ","Recording times for the
simulation")
    visibility=filemenu2.Append(ID_MENU_CollISION , "&Visibility","anylize the collision for whole
simulation")
    TOOL =filemenu2.Append(ID_MENU_TOOL , "&Number of tools'changes","anylize the number
of converting tools")
    REACH =filemenu2.Append(ID_MENU_REACH , "&SDR Angle","anylize the Accessibility of each
part")
    self.Bind(wx.EVT_MENU,self.OnPath,Tar)#选择 about 将执行 onAbout 函数
    self.Bind(wx.EVT_MENU,self.OnVisibility,visibility)


     ####菜单 3
    ID_MENU_1= wx.NewId()
    ID_MENU_2 = wx.NewId()
    ID_MENU_3 = wx.NewId()
    ID_MENU_4 = wx.NewId()
    ID_MENU_5 = wx.NewId()
    filemenu3=wx.Menu()#菜单
    T1=filemenu3.Append(ID_MENU_1,"&Next Target","All the Trajectories for all the components")
    T2=filemenu3.Append(ID_MENU_2, "&Tool1","Recording times for the simulation")
    T3=filemenu3.Append(ID_MENU_3 , "&Tool2","anylize the collision for whole simulation")
    T4=filemenu3.Append(ID_MENU_4 , "&Tool3","anylize the number of converting tools")
    T5=filemenu3.Append(ID_MENU_5 , "&Tool4","anylize the Accessibility of each part")
    self.Bind(wx.EVT_MENU,self.clear,T1)

    # 工具栏 1
    ID_MENU_6= wx.NewId()
    ID_MENU_7 = wx.NewId()
    ID_MENU_8 = wx.NewId()
    ID_MENU_9 = wx.NewId()
    ID_MENU_10 = wx.NewId()
    ID_MENU_11= wx.NewId()
    ID_MENU_12= wx.NewId()
```

```
        ID_MENU_13 = wx.NewId()
        ID_MENU_14 = wx.NewId()
        ID_MENU_15 = wx.NewId()
        toolbar1 = wx.ToolBar(self)
        sphere=toolbar1.AddLabelTool(ID_MENU_6, '', wx.Bitmap(sys.path[0]+os.sep+'pixmaps-
menu'+os.sep+'stock_no_20.png'))
        cube=toolbar1.AddLabelTool(ID_MENU_7, '',
wx.Bitmap(sys.path[0]+os.sep+'pixmaps'+os.sep+'Object-GRCUBE.png'))
        cylinder=toolbar1.AddLabelTool(ID_MENU_8, '',
wx.Bitmap(sys.path[0]+os.sep+'pixmaps'+os.sep+'Object-GRCYLINDER1.png'))
        haptic=toolbar1.AddLabelTool(ID_MENU_9, '',
wx.Bitmap(sys.path[0]+os.sep+'pixmaps'+os.sep+'ico-haptic2.png'))
        collision=toolbar1.AddLabelTool(ID_MENU_10, '',
wx.Bitmap(sys.path[0]+os.sep+'pixmaps'+os.sep+'Device-CVEODE.png'))
        D3D=toolbar1.AddLabelTool(ID_MENU_11, '',
wx.Bitmap(sys.path[0]+os.sep+'pixmaps'+os.sep+'converted.png'))
        colorChange =toolbar1.AddLabelTool(ID_MENU_12, '',
wx.Bitmap(sys.path[0]+os.sep+'pixmaps'+os.sep+'color_picker.png'))
        camera=toolbar1.AddLabelTool(ID_MENU_13, '',
wx.Bitmap(sys.path[0]+os.sep+'pixmaps'+os.sep+'camer.png'))
        screenShot=toolbar1.AddLabelTool(ID_MENU_14, '',
wx.Bitmap(sys.path[0]+os.sep+'pixmaps'+os.sep+'Screenshot.png'))
        toolbar1.AddSeparator()
        toolbar1.Realize()

        self.Bind(wx.EVT_TOOL,self.OnSphere,sphere)#
        self.Bind(wx.EVT_TOOL,self.OnCube,cube)
        self.Bind(wx.EVT_TOOL,self.OnCylinder,cylinder)
        self.Bind(wx.EVT_TOOL,self.OnHaptic,haptic)
        self.Bind(wx.EVT_TOOL,self.OnCollision,collision)
        self.Bind(wx.EVT_TOOL,self.OnD3D,D3D)
        self.Bind(wx.EVT_TOOL,self.OnOtherColor,colorChange)
        self.Bind(wx.EVT_TOOL,self.OnCamera,camera)
        self.Bind(wx.EVT_TOOL,self.OnScreenshot,screenShot)
        ID_MENU_16= wx.NewId()
        ID_MENU_17 = wx.NewId()
        ID_MENU_18 = wx.NewId()
        ID_MENU_19 = wx.NewId()
        ID_MENU_20 = wx.NewId()
        ID_MENU_21= wx.NewId()
        ID_MENU_22= wx.NewId()
        ID_MENU_23 = wx.NewId()
        ID_MENU_24 = wx.NewId()
        ID_MENU_25 = wx.NewId()
        toolbar2 = wx.ToolBar(self)
        cordinate = toolbar2.AddLabelTool(ID_MENU_16, '',
wx.Bitmap(sys.path[0]+os.sep+'pixmaps'+os.sep+'Coordin.png'))
        picture = toolbar2.AddLabelTool(ID_MENU_17, '',
wx.Bitmap(sys.path[0]+os.sep+'pixmaps'+os.sep+'Pic.png'))
        self.Bind(wx.EVT_TOOL,self.Oncordinate,cordinate)
        self.Bind(wx.EVT_TOOL,self.Onpicture,picture)
        toolbar2.AddSeparator()
```

```
        toolbar2.Realize()
        vbox = wx.BoxSizer(wx.VERTICAL)
        vbox.Add(toolbar1, 0, wx.EXPAND)
        vbox.Add(toolbar2, 100, wx.EXPAND)

        self.SetSizer(vbox)

        #~ box.Add(s, 0, wx.ALL, 10)
        #~ self.SetClientSize(self.sizer.GetSize())
        #self.SetSize((self.col*24,self.nbcol*24)
        self.Fit()

        self.Show(True)

        menuBar=wx.MenuBar()
        menuBar.Append(filemenu,"&File")
        menuBar.Append(filemenu2,"&Analysis")
        menuBar.Append(filemenu3,"&Tools")
        self.SetMenuBar(menuBar)
        self.Show(True)

    def Keypress(self, obj, event):
        key = obj.GetKeySym()
        if key == "e":
            obj.InvokeEvent("DeleteAllObjects")
            sys.exit()
        elif key == "w":
            self.Wireframe()
        elif key =="s":
            self.Surface()

    def Wireframe(self ):
        actors =self.Renderer.GetActors()
        actors.InitTraversal()
        actor = actors.GetNextItem()
        while actor:
            actor.GetProperty().SetRepresentationToWireframe()
            actor = actors.GetNextItem()
        self.RenderWindow.Render()

    def Surface(self ):
        actors =self.Renderer.GetActors()
        actors.InitTraversal()
        actor = actors.GetNextItem()
        while actor:
            actor.GetProperty().SetRepresentationToSurface()
            actor = actors.GetNextItem()
        self.RenderWindow.Render()

    def OnAbout(self,e):
        dlg=wx.MessageDialog(self,"Disassembly simulation", "About this application", wx.OK)
        dlg.ShowModal()
```

```
        dlg.Destroy()
    def OnExit(self,e):
        self.Close(True)
        sys.exit()

    def OnSave(self,e):
        dlg=wx.FileDialog(self, "save file as...", os.getcwd(), "", "
*.vtk",wx.SAVE|wx.OVERWRITE_PROMPT)
        result=dlg.ShowModal()
        inFile=dlg.GetPath()
        if result==wx.ID_OK:
            save(self,inFile)
            return True
        elif result==wx.ID_CANCEL:
            return False

    def importVrml(self,fileName):
        importer=vtk.vtkVRMLImporter()
        importer.SetFileName(fileName)
        importer.Read()
        importer.SetRenderWindow(self.RenderWindow)
        importer.Update()
        actors=importer.GetRenderer().GetActors()
        c=actors.GetNumberOfItems ()
        print(c)
        actors.InitTraversal()
        self.RenderWindow.Render()
        self.iren.Initialize()

    def importStl(self,fileList):
        for i in fileList:
            reader = vtk.vtkSTLReader()
            reader.SetFileName(i)
            mapper =vtk.vtkPolyDataMapper()
            mapper.SetInputConnection(reader.GetOutputPort())
            lxactor =vtk.vtkActor()
            lxactor.SetMapper(mapper)
            self.Renderer.AddActor(lxactor)
            self.RenderWindow.Render()
    def OnOpen(self,e):
        self.dirname=''
        dlg=wx.FileDialog(self,"choose a file",self.dirname,"","*.stl",wx.MULTIPLE)
        if dlg.ShowModal()==wx.ID_OK:
            fileList=dlg.GetPaths()
            self.importStl(fileList)
            #self.filename=dlg.GetFilename()
            #self.dirname=dlg.GetDirectory()
            #f=self.importStl(os.path.join(self.dirname,self.filename))
        dlg.Destroy()
    def OnOtherColor(self,event):
        self.dog=wx.ColourDialog(self)
        self.dog.GetColourData().SetChooseFull(True)
```

```
    if self.dog.ShowModal()== wx.ID_OK:
        colour_data = self.dog.GetColourData()
        colour = colour_data.GetColour()
        colour1=(colour[0]/255,colour[1]/255,colour[2]/255)
        print colour
        if  self.actor!= None:
            self.actor.GetProperty().SetColor(colour1)
            self.RenderWindow.Render()
        #self.pickerActor.SetColor(dlg.GetColourData().GetColour())
    self.dog.Destroy()

def OnSphere(self,e):
    cc=vtk_sphere(1,20,20,0.5,0.8,0.1)
    self.Renderer.AddActor(cc)
    self.RenderWindow.Render()

def OnCube(self,e):
    dd=vtk_Cube(1,1,1,r=0.4,g=0.8,b=0.1)
    self.Renderer.AddActor(dd)
    self.RenderWindow.Render()

def OnCylinder(self,e):
    self.ee=vtk_cylinder(1,1,20,0.1,0.5,0.2)
    self.Renderer.AddActor(self.ee)
    self.RenderWindow.Render()

def OnCamera(self,e):
    cameraAssembly=vtk_Camera_Arrow()
    self.Renderer.AddActor(cameraAssembly)
    self.RenderWindow.Render()


def OnScreenshot(self,e):
    import random
    rangeLetter=('abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ')
    newFileName = random.choice(rangeLetter)+ ".png"
    windowToImageFilter=vtk.vtkWindowToImageFilter()
    windowToImageFilter.SetInput(self.RenderWindow)
    windowToImageFilter.SetMagnification(3)
    windowToImageFilter.SetInputBufferTypeToRGBA()
    writer=vtk.vtkPNGWriter()
    writer.SetFileName(newFileName)
    writer.SetInputConnection(windowToImageFilter.GetOutputPort())
    writer.Write()
    os.system(newFileName)

def OnVisibility(self,e):
    actors=self.Renderer.GetActors()
    actors.InitTraversal()
    actor = actors.GetNextItem()
    while actor:
        if  self.actor!= actor:
```

```
            actor.GetProperty().SetColor(0,0,0)
        actor = actors.GetNextItem()
    self.Renderer.SetBackground(0,0,0)
    self.RenderWindow.Render()

def OnHaptic (self,e):
    print(self.ha.HAPapiVersion())
    self.ha.HAPsetTimeStep(0.002)#default value is 0.002f.
    self.ha.HAPsetIndexingMode(3.0)
    self.ha.HAPsetForceFactor(3.0)
    self.ha.HAPsetSpeedFactor(1.5)
    tab=[0.0,0.0,0.0,0.0,0.0,0.0,1.0]
    self.ha.HAPsetBaseFrame(tab)
    self.ha.HAPsetObservationFrameSpeed(tab)
    self.ha.HAPsetCommandType(5)#comannd type virtmech
    self.ha.HAPconnect()
    while(self.ha.isConnected==True):
        transform=position2matrix(self.ha.posCT, 20)
        if transform!=self.transform:
            self.transform=transform
            newTransform=vtk.vtkTransform()
            newTransform.SetMatrix(self.Transform)
            self.dd.SetPosition(newTransform.GetPosition())
            self.dd.SetOrientation(newTransform.GetOrientation())
            self.RenderWindow.Render()
def OnCollision (self,e):
    print('00000')

def OnD3D(self,e):
    self.RenderWindow.StereoRenderOn()
    self.RenderWindow.SetStereoTypeToInterlaced()
    self.RenderWindow.Render()
    self.iren.Start()
def clear(self,e):
    self.sphereX=[]
def OnPath(self,e):
    startPoints=[]
    endPoints=[]
    Points=[]
    color=self.Dactor.GetProperty().GetColor()
    print self.sphereX
    for i in  range (0,len(self.sphereX)-1,2) :

        Points.append((self.sphereX[i]))
    for i in range(len(Points)-1):
        startPoints.append(Points[i])
        endPoints.append(Points[i+1])
    for i in range (len(startPoints)):
        sphereStartSource = vtk.vtkSphereSource()
        lineSource=vtk.vtkLineSource()
        lineSource.SetPoint1(startPoints[i])
        lineSource.SetPoint2(endPoints[i])
```

```
        lineSource.Update()
        lineMapper=vtk.vtkPolyDataMapper()
        lineMapper.SetInputConnection(lineSource.GetOutputPort())
        lineActor=vtk.vtkActor()
        lineActor.GetProperty().SetColor(0.0,0.0,0.0)
        lineActor.SetMapper(lineMapper)
        sphereStartSource.SetCenter(startPoints[i])
        sphereStartSource.SetRadius(5)
        sphereStartMapper = vtk.vtkPolyDataMapper()
        sphereStartMapper.SetInputConnection(sphereStartSource.GetOutputPort())
        sphereStart = vtk.vtkActor()
        sphereStart.SetMapper(sphereStartMapper)
        sphereStart.GetProperty().SetColor(color)
        self.Renderer.AddActor(sphereStart)
        self.Renderer.AddActor(lineActor)
        self.RenderWindow.Render()

##############################################################################The
own interactor style!
  def ButtonEvent(self,Self,event):
    global Rotating, Panning, cameraRotating
    if event == "LeftButtonPressEvent":
      self.Rotating = 1
      XYpos=self.iren.GetEventPosition()
      x= XYpos[0]
      y= XYpos[1]
      CurrentRenderer = self.iren.FindPokedRenderer(x,y)
      picker=vtk.vtkPropPicker()
      picker.Pick(x, y,0,CurrentRenderer)
      self.actor=picker.GetActor()
    elif event == "LeftButtonReleaseEvent":
      self.Rotating = 0
    elif event == "MiddleButtonPressEvent":
      self.Panning = 1
    elif event == "MiddleButtonReleaseEvent":
      self.Panning = 0
    elif event == "RightButtonPressEvent":
      self.cameraRotating = 1
    elif event == "RightButtonReleaseEvent":
      self.cameraRotating = 0
    elif event == "MouseWheelForwardEvent":
      self.zooming()
    elif event == "MouseWheelBackwardEvent":
      self.shrinking()

  def MouseMove(self,obj, event):
    global Rotating, Panning, Zooming
    global iren, renWin, ren

    if self.Rotating:
      self.Rotation()
    if self.Panning:
```

```
            self.pan()
        elif self.cameraRotating:
            self.cameraRotation()

    def pan(self):

        XYpos=self.iren.GetEventPosition()
        x= XYpos[0]
        y= XYpos[1]
        LastXYpos =self.iren.GetLastEventPosition()
        CurrentRenderer = self.iren.FindPokedRenderer(x,y)
        picker=vtk.vtkPropPicker()
        picker.Pick(x, y,0,CurrentRenderer)
        self.Dactor=picker.GetActor()
        if (self.Dactor==None):
            self.Dactor=picker.GetAssembly()
            if(self.Dactor==None):
                return
        self.RenderWindow.Render()
        def startPan():
            disp_obj_center =[]
            new_pick_point =[]
            old_pick_point =[]
            motion_vector  =[]
            obj_center=self.Dactor.GetCenter()
            self.sphereX.append(obj_center)
            CurrentRenderer.SetWorldPoint(obj_center[0],obj_center[1],obj_center[2],1.0)
            CurrentRenderer.WorldToView()
            display=CurrentRenderer.GetViewPoint()
            CurrentRenderer.ViewToDisplay()
            isp_obj_center=CurrentRenderer.GetDisplayPoint()
            CurrentRenderer.SetDisplayPoint(XYpos[0],XYpos[1],0)
            CurrentRenderer.DisplayToView()
            CurrentRenderer.GetViewPoint()
            CurrentRenderer.ViewToWorld()
            new_pick_point=CurrentRenderer.GetWorldPoint()
            CurrentRenderer.SetDisplayPoint(LastXYpos[0],LastXYpos[1],0)
            CurrentRenderer.DisplayToView()
            CurrentRenderer.GetViewPoint()
            CurrentRenderer.ViewToWorld()
            old_pick_point=CurrentRenderer.GetWorldPoint()
            motion_vector.append(new_pick_point[0] - old_pick_point[0])
            motion_vector.append(new_pick_point[1] - old_pick_point[1])
            motion_vector.append(new_pick_point[2] - old_pick_point[2])
            if (self.Dactor.GetUserMatrix()!=None):
                t=vtk.vtkTransform()
                t.PostMultiply()
                t.SetMatrix(Dactor.GetUserMatrix())
                t.Translate(motion_vector[0]*20,motion_vector[1]*20,motion_vector[2]*20)
                self.Dactor.GetUserMatrix().DeepCopy(t.GetMatrix())
                self.Dactor.AddPosition(motion_vector[0],motion_vector[1],motion_vector[2])
            else:
```

```
            self.Dactor.AddPosition(motion_vector[0]*2,motion_vector[1]*2,motion_vector[2]*2)
        CurrentRenderer.ResetCameraClippingRange()
        #collision_detection(actors,n=len(actors),density=2000,r=0.5)
        self.RenderWindow.Render()
    startPan()

def Rotation(self):
    XYpos=self.iren.GetEventPosition()
    x= XYpos[0]
    y= XYpos[1]
    LastXYpos =self.iren.GetLastEventPosition()
    CurrentRenderer = self.iren.FindPokedRenderer(x,y)
    picker=vtk.vtkPropPicker()
    picker.Pick(x, y,0,CurrentRenderer)
    Dactor=picker.GetActor()
    if (Dactor==None):
        Dactor=picker.GetAssembly()
        if(Dactor==None):
            return
    self.RenderWindow.Render()
    def startRotation():
        cam=self.Renderer.GetActiveCamera()
        center=Dactor.GetCenter()
        boundRadius=Dactor.GetLength()*0.5#half of the diagonal of the bounding box
        cam.OrthogonalizeViewUp()#force the viewup to be perpendicular to camera->focalpoint
vector
        cam.ComputeViewPlaneNormal()
        view_up=list(cam.GetViewUp())
        c=vtk.vtkMath()
        c.Normalize(view_up)#Unit Vectors
        view_look=cam.GetViewPlaneNormal()
        view_right=[0.0,0.0,0.0]
        c.Cross(view_up,view_look,view_right)
        c.Normalize(view_right)#Unit Vectors
        outsidept=[]
        outsidept.append(center[0] + view_right[0] * boundRadius)
        outsidept.append(center[1] + view_right[1] * boundRadius)
        outsidept.append(center[2] + view_right[2] * boundRadius)#pas compris
        CurrentRenderer.SetWorldPoint(center[0],center[1],center[2],1.0)
        CurrentRenderer.WorldToView()
        CurrentRenderer.GetViewPoint()
        CurrentRenderer.ViewToDisplay()
        disp_obj_center=CurrentRenderer.GetDisplayPoint()
        CurrentRenderer.SetWorldPoint(outsidept[0],outsidept[1],outsidept[2],1.0)
        CurrentRenderer.WorldToView()
        CurrentRenderer.GetViewPoint()
        CurrentRenderer.ViewToDisplay()
        outsidept=CurrentRenderer.GetDisplayPoint()
        radius=math.sqrt(c.Distance2BetweenPoints(disp_obj_center,outsidept))
        nxf=(x-disp_obj_center[0])/radius
        nyf=(y-disp_obj_center[1])/radius
        oxf=(LastXYpos[0]-disp_obj_center[0])/radius
```

```
        oyf=(LastXYpos[1]-disp_obj_center[1])/radius

        cc=nxf * nxf + nyf * nyf
        dd=oxf * oxf + oyf * oyf

        if (((nxf * nxf + nyf * nyf) <= 1.0)and((oxf * oxf + oyf * oyf) <= 1.0)):
            newXAngle = c.DegreesFromRadians( math.sin( nxf ) )
            newYAngle = c.DegreesFromRadians( math.sin( nyf ) )
            oldXAngle = c.DegreesFromRadians( math.sin( oxf ) )
            oldYAngle = c.DegreesFromRadians( math.sin( oyf ) )
            scale=[1.0,1.0,1.0]
            rotate=[[0.0,0.0,0.0,0.0,0.0], [ 0.0,0.0,0.0,0.0,0.0]]
             rotate[0][0] = newXAngle - oldXAngle
            rotate[0][1] = view_up[0]
            rotate[0][2] = view_up[1]
            rotate[0][3] = view_up[2]
                rotate[1][0] = oldYAngle - newYAngle
            rotate[1][1] = view_right[0]
            rotate[1][2] = view_right[1]
            rotate[1][3] = view_right[2]
            self.Prop3DTransform(Dactor,center,2,rotate,scale)
           #collision_detection(actors,n=len(actors),density=2000,r=0.5)
            self.RenderWindow.Render()
    startRotation()
def Prop3DTransform(self,actor,center,Num,rotate,scale):
     oldM=actor.GetMatrix()
    orig=actor.GetOrigin()
    newTransform=vtk.vtkTransform()
    newTransform.PostMultiply()
    if(actor.GetUserMatrix()!=None):
        newTransform.SetMatrix(actor.GetUserMatrix())
    else:
        newTransform.SetMatrix(oldM)
          newTransform.Translate(-center[0],-center[1],-center[2])
        for i in range(Num):
     newTransform.RotateWXYZ(rotate[i][0], rotate[i][1],rotate[i][2], rotate[i][3])
    newTransform.Translate(center[0],center[1], center[2])
    newTransform.Translate(-(orig[0]), -(orig[1]), -(orig[2]))
    newTransform.PreMultiply()
    newTransform.Translate(orig[0], orig[1], orig[2])
    if (actor.GetUserMatrix() != None):
        newTransform.GetMatrix(actor.GetUserMatrix())
    else:
        actor.SetPosition(newTransform.GetPosition())
        actor.SetScale(newTransform.GetScale())
        actor.SetOrientation(newTransform.GetOrientation())
    newTransform.Pop()
def cameraRotation(self):
    XYpos=self.iren.GetEventPosition()
    LastXYpos =self.iren.GetLastEventPosition()
```

```
        x= XYpos[0]
        y= XYpos[1]
        dx=XYpos[0]-LastXYpos[0]
        dy=XYpos[1]-LastXYpos[1]
        CurrentRenderer = self.iren.FindPokedRenderer(x,y)
        size=CurrentRenderer.GetRenderWindow().GetSize()
        delta_elevation = -20.0 / size[1] * 10
        delta_azimuth = -20.0 / size[0]* 10
        rxf = dx * delta_azimuth
        ryf = dy * delta_elevation
        camera =CurrentRenderer.GetActiveCamera()
        camera.Azimuth(rxf)
        camera.Elevation(ryf)
        camera.OrthogonalizeViewUp()
        CurrentRenderer.ResetCameraClippingRange()
        self.RenderWindow.Render()
    def zooming(self):
        XYpos=self.iren.GetEventPosition()
        LastXYpos = self.iren.GetLastEventPosition()
        x= XYpos[0]
        y= XYpos[1]
        dy=XYpos[1]-LastXYpos[1]
        CurrentRenderer = self.iren.FindPokedRenderer(x,y)
        center= CurrentRenderer.GetCenter()
        factor=pow(1.1,1)
        print(factor)
        camera = CurrentRenderer.GetActiveCamera()
        if camera.GetParallelProjection():
            camera.SetParallelScale(camera.GetParallelScale()/factor)
        else:
            camera.Dolly(factor)
        CurrentRenderer.ResetCameraClippingRange()
        self.RenderWindow.Render()

    def shrinking(self):
        XYpos=self.iren.GetEventPosition()
        LastXYpos =self.iren.GetLastEventPosition()
        x= XYpos[0]
        y= XYpos[1]
        CurrentRenderer = self.iren.FindPokedRenderer(x,y)
        center= CurrentRenderer.GetCenter()
        factor=pow(1.1,-1)
        camera = CurrentRenderer.GetActiveCamera()

        if camera.GetParallelProjection():
            camera.SetParallelScale(camera.GetParallelScale()/factor)
        else:
            camera.Dolly(factor)
        CurrentRenderer.ResetCameraClippingRange()
        self.RenderWindow.Render()

    def Button(self,text='',filename=None,bhelp='',Toggle=False,Popup=False):
```

```
        PUSH=True
        tdir=sys.path[0]
        if filename :
            if not Toggle : item = wx.BitmapButton(self, -1, wx.Bitmap(tdir+os.sep+'pixmaps-
menu'+os.sep+filename))
            else :
                bmp=wx.Bitmap(tdir+os.sep+'pixmaps-menu'+os.sep+filename)
                item = buttons.GenBitmapToggleButton(self, -1, bmp)
        else :
            if not Toggle :
                item = wx.Button( self, 10000, text, wx.DefaultPosition, wx.DefaultSize )
            else :
                item = wx.ToggleButton( self, 10000, text, wx.DefaultPosition, wx.DefaultSize )
        self.sizer.Add( item ,  (self.row,self.col))
        self.row+=1
        if self.row>=self.nbcol :
            self.col+=1
            self.row=0
        return item

    def Oncordinate(self,e):
        axes=vtk.vtkAxesActor()
        widget=vtk.vtkOrientationMarkerWidget()
        widget.SetOutlineColor( 0.9300, 0.5700, 0.1300 )
        widget.SetOrientationMarker( axes )
        widget.SetInteractor(self.iren)
        widget.SetViewport(0.0, 0.0, 0.4, .4)
        widget.SetEnabled(1)
        widget.InteractiveOn()
        self.Renderer.ResetCamera()
        self.RenderWindow.Render()
    def Onpicture(self,e):
        dir= os.getcwd()
        initial_dir=os.path.join(dir,'png')
        dlg=ib.ImageDialog(self,initial_dir)
        dlg.Centre()
        if dlg.ShowModal() == wx.ID_OK:
            # show the selected file
            areaReading.areaRead(dlg.GetFile())
        else:
            print "You pressed Cancel"
        dlg.Destroy()

app=wx.App(False)
frame=myFrame(None,"Disassembly simulation")
app.MainLoop()
```

Appendix B collision detection with ODE

```
import vtk
import ode
import threading
import sys
import time
import math

def vtk_sphere(radius,theta,phi,r,g,b):
   sphereSource=vtk.vtkSphereSource()
   sphereSource.SetRadius(radius)
   sphereSource.SetThetaResolution (theta)
   sphereSource.SetPhiResolution (phi)
   actor1=vtk.vtkActor()
   mapper=vtk.vtkPolyDataMapper()
   mapper.SetInputConnection(sphereSource.GetOutputPort())
   actor1.SetMapper(mapper)
   actor1.GetProperty().SetColor (r,g,b)
   return actor1

def vtk_Cube(x,y,z,r=0.5,g=0.5,b=0.5):
   cyl=vtk.vtkCubeSource()
   cyl.SetXLength(x)
   cyl.SetYLength(y)
   cyl.SetZLength(z)
   mapper =vtk.vtkPolyDataMapper ()
   mapper.SetInputConnection(cyl.GetOutputPort())
   lxactor =vtk.vtkActor()
   lxactor.SetMapper(mapper)
   lxactor.GetProperty().SetColor (r, g, b)
   return lxactor

def vtk_cylinder(radius,h,resolution,r,g,b):
   cy=vtk.vtkCylinderSource()
   cy.SetRadius(radius)
   cy.SetHeight(h)
   cy.SetResolution(resolution)


   translation =vtk.vtkTransform()
   translation.RotateX (90.0)
   transformFilter =vtk.vtkTransformPolyDataFilter()
   transformFilter.SetInputConnection(cy.GetOutputPort())
   transformFilter.SetTransform(translation)
   mapper =vtk.vtkPolyDataMapper()
   mapper.SetInputConnection(transformFilter.GetOutputPort())
   lxactor =vtk.vtkActor()
   lxactor.SetMapper(mapper)
```

```
    lxactor.GetProperty().SetColor (r, g, b)
    return lxactor


    #meshdata = ode.TriMeshData()  #create the data buffer
    #meshdata.build(verts,faces)  #Put vertex and face data into the buffer
    #mesh = ode.GeomTriMesh(meshdata,myspace) #

#vtk models
sources=[]
actors=[]
actor1 = vtk_sphere(0.5,20,20,1.0,0.5,0)
actor2= vtk_Cube(0.5,0.5,1,r=1.0,g=0.5,b=0.5)
actor3 = vtk_cylinder(0.5,1,20,0,0.8,1.0)

c=vtk.vtkVersion()
print(c.GetVTKVersion())

actor4= vtk_Cube(0.5,0.5,1,r=0.0,g=1.0,b=0.5)

actor1.SetPosition(0,5,0)
actor2.SetPosition(0,3,0)
actor3.SetPosition(3,3,0)
actor4.SetPosition(0,0,0)

cy=vtk_cylinder(0.8,1.0,20,0.8,0.2,0.3)
cy1=vtk_cylinder(0.5,1.0,20,0.8,0.2,0.4)
cy.SetPosition(0,0,-0.5)
cy1.SetPosition(0,0,0.5)
ass=vtk.vtkAssembly()
ass.AddPart(cy)
ass.AddPart(cy1)
ass.SetPosition(0,2,0)
bounds=ass.GetBounds()
size=[bounds[1]-bounds[0],bounds[3]-bounds[2],bounds[5]-bounds[4]]
# define sources for later to use
actors.append(actor1)
actors.append(actor2)
actors.append(actor3)
actors.append(actor4)
actors.append(ass)
renderer=vtk.vtkRenderer()
renderer.SetBackground(2/3, 1/2, 1)
renderer.AddActor(actor1)
renderer.AddActor(actor2)
renderer.AddActor(actor3)
renderer.AddActor(actor4)
renderer.AddActor(ass)
renWin=vtk.vtkRenderWindow()
renWin.SetSize(600,300)
xt=0.005
renWin.AddRenderer(renderer)
```

```
#renWin.StereoCapableWindowOn()
#renWin.StereoRenderOn()
#renWin.SetStereoTypeToInterlaced()
Rotating = 0
Panning = 0
cameraRotating = 0
Num=0
lasttime=0

iren=vtk.vtkRenderWindowInteractor()
iren.SetInteractorStyle(None)
iren.SetRenderWindow(renWin)
#define myown interaction style, still not finished, need to difine the other
#it can be used to drag and rotate one piece.
def ButtonEvent(Self,event):
   global Rotating, Panning, cameraRotating
   if event == "LeftButtonPressEvent":
      Rotating = 1
   elif event == "LeftButtonReleaseEvent":
      Rotating = 0
   elif event == "MiddleButtonPressEvent":
      Panning = 1
   elif event == "MiddleButtonReleaseEvent":
      Panning = 0
   elif event == "RightButtonPressEvent":
      cameraRotating = 1
   elif event == "RightButtonReleaseEvent":
      cameraRotating = 0
   elif event == "MouseWheelForwardEvent":
      print("forward")
      zooming()
   elif event == "MouseWheelBackwardEvent":
      print("backward")
      shrinking()
def MouseMove(obj, event):
   global Rotating, Panning, Zooming
   global iren, renWin, ren
   if Rotating:
      Rotation()
   elif Panning:
      pan()
   elif cameraRotating:
      print(cameraRotating)
      cameraRotation()
def Keypress(obj, event):
   key = obj.GetKeySym()
   if key == "e":
      obj.InvokeEvent("DeleteAllObjects")
      sys.exit()
   elif key == "w":
      Wireframe()
   elif key =="s":
```

```
      Surface()
   elif key =="c":
      addCube()

   elif key=="d":
      addSphere()

def addCube():
   actor= vtk_Cube(0.5,0.5,1,r=1.0,g=0.5,b=0.5)
   actors.append(actor)
   actor.SetPosition(0,0,0)
   renderer.AddActor(actor)
   renWin.Render()


def addSphere():
   actor = vtk_sphere(0.5,20,20,1.0,0.5,0)
   actors.append(actor)
   actor.SetPosition(0,0,0)
   renderer.AddActor(actor)
   renWin.Render()

def Wireframe():
   actors = renderer.GetActors()
   actors.InitTraversal()
   actor = actors.GetNextItem()
   while actor:
      actor.GetProperty().SetRepresentationToWireframe()
      actor = actors.GetNextItem()
   renWin.Render()

def Surface():
   actors = renderer.GetActors()
   actors.InitTraversal()
   actor = actors.GetNextItem()
   while actor:
      actor.GetProperty().SetRepresentationToSurface()
      actor = actors.GetNextItem()
   renWin.Render()

#collision detection function
def vtk_ode(mat):
   position=(mat.GetElement(0,3), mat.GetElement(1,3),mat.GetElement(2,3))
   rotation=(mat.GetElement(0,0), mat.GetElement(0,1),mat.GetElement(0,2),mat.GetElement(1,0),
mat.GetElement(1,1),mat.GetElement(1,2),mat.GetElement(2,1),
mat.GetElement(2,2),mat.GetElement(2,3))
   return position,rotation

def ode_vtk(position,rotation):
   rot=[rotation[0],rotation[3],rotation[6],0.0,
       rotation[1],rotation[4],rotation[7],0.0,
       rotation[2],rotation[5],rotation[8],0.0,
```

```
      position[0],position[1],position[2],1.0]
    return rot


def collision_detection(actorss,n,density=2000,r=0.5):
    global Dactor,actors,lasttime
    world=ode.World()
    world.setGravity((0,-9.8,0))
    world.setERP(0.8)
    world.setCFM(1E-8)
    contactgroup = ode.JointGroup()
    space=ode.Space()
    total_time=2
    dt=0.1
    body=[]
    geom=[]
    #body.setKinematic()
    for i in range(n):
        body.append(ode.Body(world))
        mass=ode.Mass()
        if (actorss[i].GetClassName()=="vtkAssembly"):
            mat=actorss[i].GetMatrix()
            vv=actorss[i].GetPosition()
            mm=actorss[i].GetCenter()

            position,rotation=vtk_ode(mat)

            body[i].setPosition(position)
            body[i].setRotation(rotation)
            body[i].setKinematic()
            body[i].boxsize = (size[0],size[1],size[2])

            geom2=ode.GeomCylinder(None,0.8,1.0)
            geom1=ode.GeomCylinder(None,0.5,1.0)
            trans=ode.GeomTransform(space)
            trans1=ode.GeomTransform(space)
            trans.setGeom(geom2)
            trans1.setGeom(geom1)
            geom2.setPosition((0,0,-0.5))
            geom1.setPosition((0,0,0.5))
            geom.append(trans)

            geom[i].setBody(body[i])
            trans1.setBody(body[i])
            if Dactor==actorss[i]:
                mass.setBox(2000,size[0],size[1],size[2])
                body[i].setMass(mass)
        else:
            algo=actorss[i].GetMapper().GetInputConnection(0,0).GetProducer()
            cc=algo.GetClassName()
            if (cc=="vtkSphereSource"):#if it is the pick one. We need to set mass to it
                mat=actorss[i].GetMatrix()
```

```
            position,rotation=vtk_ode(mat)

            body[i].setPosition(position)
            body[i].setRotation(rotation)

            body[i].setKinematic()
            geom1=ode.GeomSphere(space,0.5)
            geom.append(geom1)
            geom[i].setBody(body[i])
            if Dactor==actorss[i]:
              mass.setSphere(2000,r)
              body[i].setMass(mass)

        if (cc=="vtkCubeSource"):
            mat=actorss[i].GetMatrix()
            position,rotation=vtk_ode(mat)
            body[i].boxsize = (0.5,0.5,1)
            body[i].setPosition(position)
            body[i].setRotation(rotation)
            body[i].setKinematic()
            geom1=ode.GeomBox(space,lengths=body[i].boxsize)
            geom.append(geom1)
            geom[i].setBody(body[i])
            if Dactor==actorss[i]:
                mass.setBox(2000,0.5,0.5,1)
                body[i].setMass(mass)

        if (cc=="vtkTransformPolyDataFilter"):
            mat=actorss[i].GetMatrix()
            position,rotation=vtk_ode(mat)
            body[i].setKinematic()
            body[i].setPosition(position)
            body[i].setRotation(rotation)
            geom1=ode.GeomCylinder(space,0.5,1.0)
            geom.append(geom1)
            geom[i].setBody(body[i])
            if Dactor==actorss[i]:
              mass.setCylinder(2000,2,0.5,2.0)
              body[i].setMass(mass)

def near_callback(args, geom1, geom2):
    contacts = ode.collide(geom1, geom2)
    world,contactgroup = args
    for c in contacts:
        c.setBounce(1)
        c.setMu(10000)
        j = ode.ContactJoint(world, contactgroup, c)
        j.attach(geom1.getBody(),geom2.getBody())

def simloop():
  simstep=0.001
  space.collide((world,contactgroup), near_callback)
```

```
    world.step(simstep)
    for i in range(n):
      x,y,z=body[i].getPosition()
      actorss[i].SetPosition(x,y,z)#when the loop beginning,it is not possible to change the
#model rotation if no collision happening. For ode, the body is changing the
      renWin.Render()           #rotation happens when the world is begining.                    #The
problem is when the rotation will be feedback rotation.
      contactgroup.empty()
  dt=0.005
  while(dt<0.1):
    simloop()
    dt+=dt
#my own interaction style for disassembly products
##########################################################################################
#######
def pan():
  global Rotating, Panning, Zooming,n,Num,Dactor,lasttime
  global iren, renWin, ren, Dactor, CurrentRenderer,XYpos, LastXYpos
  XYpos=iren.GetEventPosition()
  x= XYpos[0]
  y= XYpos[1]
  LastXYpos =iren.GetLastEventPosition()

  CurrentRenderer = iren.FindPokedRenderer(x,y)
  picker=vtk.vtkPropPicker()
  picker.Pick(x, y,0,CurrentRenderer)
  Dactor=picker.GetActor()
  if (Dactor==None):
    Dactor=picker.GetAssembly()
    if(Dactor==None):
      return
  iren.Initialize()
  renWin.Render()
  def startPan():
    disp_obj_center =[]
    new_pick_point =[]
    old_pick_point =[]
    motion_vector  =[]

    global Dactor, XYpos,LastXYpos,renWin
    obj_center=Dactor.GetCenter()
    CurrentRenderer.SetWorldPoint(obj_center[0],obj_center[1],obj_center[2],1.0)
    CurrentRenderer.WorldToView()
    display=CurrentRenderer.GetViewPoint()
    CurrentRenderer.ViewToDisplay()
    isp_obj_center=CurrentRenderer.GetDisplayPoint()
    CurrentRenderer.SetDisplayPoint(XYpos[0],XYpos[1],0)
    CurrentRenderer.DisplayToView()
    CurrentRenderer.GetViewPoint()
    CurrentRenderer.ViewToWorld()
    new_pick_point=CurrentRenderer.GetWorldPoint()
    CurrentRenderer.SetDisplayPoint(LastXYpos[0],LastXYpos[1],0)
```

```
    CurrentRenderer.DisplayToView()
    CurrentRenderer.GetViewPoint()
    CurrentRenderer.ViewToWorld()
    old_pick_point=CurrentRenderer.GetWorldPoint()


    motion_vector.append(new_pick_point[0] - old_pick_point[0])
    motion_vector.append(new_pick_point[1] - old_pick_point[1])
    motion_vector.append(new_pick_point[2] - old_pick_point[2])

    if (Dactor.GetUserMatrix()!=None):
      t=vtk.vtkTransform()
      t.PostMultiply()
      t.SetMatrix(Dactor.GetUserMatrix())
      t.Translate(motion_vector[0]*10,motion_vector[1]*10,motion_vector[2]*10)
      Dactor.GetUserMatrix().DeepCopy(t.GetMatrix())

      print(t)
      Dactor.AddPosition(motion_vector[0],motion_vector[1],motion_vector[2])
    else:
      Dactor.AddPosition(motion_vector[0]*2,motion_vector[1]*2,motion_vector[2]*2)
    CurrentRenderer.ResetCameraClippingRange()
    collision_detection(actors,n=len(actors),density=2000,r=0.5)
    renWin.Render()
  startPan()

def Rotation():

  XYpos=iren.GetEventPosition()
  x= XYpos[0]
  y= XYpos[1]
  LastXYpos =iren.GetLastEventPosition()
  CurrentRenderer = iren.FindPokedRenderer(x,y)
  picker=vtk.vtkPropPicker()
  picker.Pick(x, y,0,CurrentRenderer)
  Dactor=picker.GetActor()
  if (Dactor==None):
    Dactor=picker.GetAssembly()
    if(Dactor==None):
      return
  iren.Initialize()
  renWin.Render()

  def startRotation():
    cam=renderer.GetActiveCamera()
    center=Dactor.GetCenter()
    boundRadius=Dactor.GetLength()*0.5#half of the diagonal of the bounding box
    cam.OrthogonalizeViewUp()#force the viewup to be perpendicular to camera->focalpoint vector
    cam.ComputeViewPlaneNormal()
    view_up=cam.GetViewUp()
    c=vtk.vtkMath()
    c.Normalize(view_up)#Unit Vectors
```

```
view_look=cam.GetViewPlaneNormal()

view_right=[0.0,0.0,0.0]
c.Cross(view_up,view_look,view_right)
c.Normalize(view_right)#Unit Vectors

outsidept=[]
outsidept.append(center[0] + view_right[0] * boundRadius)
outsidept.append(center[1] + view_right[1] * boundRadius)
outsidept.append(center[2] + view_right[2] * boundRadius)#pas compris

CurrentRenderer.SetWorldPoint(center[0],center[1],center[2],1.0)
CurrentRenderer.WorldToView()
CurrentRenderer.GetViewPoint()
CurrentRenderer.ViewToDisplay()
disp_obj_center=CurrentRenderer.GetDisplayPoint()

CurrentRenderer.SetWorldPoint(outsidept[0],outsidept[1],outsidept[2],1.0)
CurrentRenderer.WorldToView()
CurrentRenderer.GetViewPoint()
CurrentRenderer.ViewToDisplay()
outsidept=CurrentRenderer.GetDisplayPoint()

radius=math.sqrt(c.Distance2BetweenPoints(disp_obj_center,outsidept))
print(radius)
nxf=(x-disp_obj_center[0])/radius
nyf=(y-disp_obj_center[1])/radius
print(x,y)
oxf=(LastXYpos[0]-disp_obj_center[0])/radius
oyf=(LastXYpos[1]-disp_obj_center[1])/radius

print(LastXYpos[0], LastXYpos[1])

cc=nxf * nxf + nyf * nyf
dd=oxf * oxf + oyf * oyf
print(dd)
if (((nxf * nxf + nyf * nyf) <= 1.0)and((oxf * oxf + oyf * oyf) <= 1.0)):
    newXAngle = c.DegreesFromRadians( math.sin( nxf ) )
    newYAngle = c.DegreesFromRadians( math.sin( nyf ) )
    oldXAngle = c.DegreesFromRadians( math.sin( oxf ) )
    oldYAngle = c.DegreesFromRadians( math.sin( oyf ) )
    scale=[1.0,1.0,1.0]
    rotate=[[0.0,0.0,0.0,0.0,0.0], [ 0.0,0.0,0.0,0.0,0.0]]

    rotate[0][0] = newXAngle - oldXAngle
    rotate[0][1] = view_up[0]
    rotate[0][2] = view_up[1]
    rotate[0][3] = view_up[2]

    rotate[1][0] = oldYAngle - newYAngle
    rotate[1][1] = view_right[0]
    rotate[1][2] = view_right[1]
```

```
            rotate[1][3] = view_right[2]
            Prop3DTransform(Dactor,center,2,rotate,scale)
        collision_detection(actors,n=len(actors),density=2000,r=0.5)
        renWin.Render()
    startRotation()


def Prop3DTransform(actor,center,Num,rotate,scale):

    print("good")

    oldM=actor.GetMatrix()
    orig=actor.GetOrigin()
    newTransform=vtk.vtkTransform()
    newTransform.PostMultiply()
    if(actor.GetUserMatrix()!=None):
        newTransform.SetMatrix(actor.GetUserMatrix())
    else:
        newTransform.SetMatrix(oldM)

    newTransform.Translate(-center[0],-center[1],-center[2])

    for i in range(Num):

        newTransform.RotateWXYZ(rotate[i][0], rotate[i][1],rotate[i][2], rotate[i][3])

    newTransform.Translate(center[0],center[1], center[2])
    newTransform.Translate(-(orig[0]), -(orig[1]), -(orig[2]))
    newTransform.PreMultiply()
    newTransform.Translate(orig[0], orig[1], orig[2])
    if (actor.GetUserMatrix() != None):
        newTransform.GetMatrix(actor.GetUserMatrix())
    else:
        actor.SetPosition(newTransform.GetPosition())
        actor.SetScale(newTransform.GetScale())
        actor.SetOrientation(newTransform.GetOrientation())
    newTransform.Pop()

def cameraRotation():

    XYpos=iren.GetEventPosition()
    LastXYpos =iren.GetLastEventPosition()
    x= XYpos[0]
    y= XYpos[1]
    dx=XYpos[0]-LastXYpos[0]
    dy=XYpos[1]-LastXYpos[1]
    CurrentRenderer = iren.FindPokedRenderer(x,y)
    size=CurrentRenderer.GetRenderWindow().GetSize()
    delta_elevation = -20.0 / size[1] * 10
    delta_azimuth = -20.0 / size[0]* 10
    rxf = dx * delta_azimuth
    ryf = dy * delta_elevation
```

```
    camera =CurrentRenderer.GetActiveCamera()
    camera.Azimuth(rxf);
    camera.Elevation(ryf);
    camera.OrthogonalizeViewUp()
    CurrentRenderer.ResetCameraClippingRange()
    renWin.Render()

def zooming():
    XYpos=iren.GetEventPosition()
    LastXYpos =iren.GetLastEventPosition()
    x= XYpos[0]
    y= XYpos[1]
    dy=XYpos[1]-LastXYpos[1]
    CurrentRenderer = iren.FindPokedRenderer(x,y)
    center= CurrentRenderer.GetCenter()
    factor=pow(1.1,1)
    print(factor)
    camera = CurrentRenderer.GetActiveCamera()
    if camera.GetParallelProjection():
        camera.SetParallelScale(camera.GetParallelScale()/factor)
    else:
        camera.Dolly(factor)
    CurrentRenderer.ResetCameraClippingRange()
    renWin.Render()

def shrinking():

    XYpos=iren.GetEventPosition()
    LastXYpos =iren.GetLastEventPosition()
    x= XYpos[0]
    y= XYpos[1]
    CurrentRenderer = iren.FindPokedRenderer(x,y)
    center= CurrentRenderer.GetCenter()

    factor=pow(1.1,-1)

    camera = CurrentRenderer.GetActiveCamera()

    if camera.GetParallelProjection():
        camera.SetParallelScale(camera.GetParallelScale()/factor)
    else:
        camera.Dolly(factor)

    CurrentRenderer.ResetCameraClippingRange()
    renWin.Render()


iren.AddObserver("LeftButtonPressEvent", ButtonEvent)
iren.AddObserver("LeftButtonReleaseEvent", ButtonEvent)
iren.AddObserver("MiddleButtonPressEvent", ButtonEvent)
iren.AddObserver("MiddleButtonReleaseEvent", ButtonEvent)
iren.AddObserver("RightButtonPressEvent", ButtonEvent)
```

```
iren.AddObserver("RightButtonReleaseEvent", ButtonEvent)
iren.AddObserver("MouseWheelForwardEvent", ButtonEvent)
iren.AddObserver("MouseWheelBackwardEvent", ButtonEvent)
iren.AddObserver("MouseMoveEvent", MouseMove)
iren.AddObserver("KeyPressEvent", Keypress)

iren.Initialize()
renWin.Render()
iren.Start()
```

Summary:
Integration of disassembly operations during product design is an important issue today. It is estimated that at the earliest stages of product design, the cost of disassembly operations almost represents 30 % of its total cost. Nowadays, disassembly operation simulation of industrial products finds a strong interest in interactive simulations through immersive and real-time schemes. In this context, in the first place, this thesis presents a method for generating the feasible disassembly sequences for selective disassembly. The method is based on the lowest levels of a disassembly product graph. Instead of considering the geometric constraints for each pair of components, the proposed method considers the geometric contact and collision relationships among the components in order to generate the so-called Disassembly Geometry Contacting Graph (DGCG). The latter is then used for disassembly sequence generation thus allowing the number of possible sequences to be reduced by ignoring any components which are unrelated to the target. A simulation framework was developed integrated in a Virtual reality environment thus allowing generating the minimum number of possible disassembly sequences. Secondly, a method for disassembly operation evaluation by 3D geometric removability analysis in a Virtual environment is proposed. It is based on seven new criteria which are: *visibility of a part*, *disassembly angles*, *number of tools' changes*, *path orientation changing*, *sub-assembly stability*, n*eck score* and *bending score*. All criteria are presented by dimensionless coefficients automatically calculated, thus allowing evaluating disassembly sequences complexity. For this purpose, a mixed virtual reality disassembly environment (VRDE) is developed based on Python programming language, utilizing VTK (Visualization Toolkit) and ODE (Open Dynamics Engine) libraries. The framework is based on *STEP*, *WRL* and *STL* exchange formats. The analysis results and findings demonstrate the feasibility of the proposed approach thus providing significant assistance for the evaluation of disassembly sequences during Product Development Process (PDP). Further consequences of the present work consist in ranking the criteria according to their importance. For this purpose, moderation coefficients may be allocated to each of them thus allowing a more comprehensive evaluating method.

Key words: disassembly sequences, disassembly product graph, geometric analysis, removability analysis, virtual reality.

Résumé:
De nos jours, l'intégration des opérations de désassemblage lors de la conception des produits est un enjeu crucial. On estime que dans la phase initiale de la conception d'un produit, le coût des opérations de désassemblage représente environ 30% de son coût total. Ainsi, la simulation des opérations de désassemblage de produits industriels trouve un fort intérêt pour des simulations interactives grâce à des programmes d'immersion et en temps réel. Dans ce contexte, dans un premier temps, cette thèse présente une méthode de génération des séquences de désassemblage possibles pour le désassemblage sélectif. La méthode est basée sur les niveaux les plus bas du graphe de désassemblage des produits. Au lieu de considérer les contraintes géométriques pour chaque paire de composants, la méthode proposée tient compte des contacts (relations géométriques entre les composants) et des collisions afin de générer le Graphe Géométrique de Contacts et de Désassemblage (DGCG). Celui-ci est ensuite utilisé pour la génération des séquences de désassemblage permettant ainsi de réduite le nombre de séquences possibles en ignorant les composants non liés avec la cible. Une application de simulation a été développée, intégrée dans un environnement de réalité virtuelle (RV) permettant ainsi la génération du nombre minimum de séquences possibles de désassemblage.

Dans un second temps, une méthode d'évaluation des opérations de désassemblage par analyse géométrique 3D de l'amovibilité dans un environnement RV est proposée. Elle est basée sur sept nouveaux critères qui sont: la visibilité d'une pièce, les angles de désassemblage, le nombre des changements d'outils, le changement d'orientation des trajectoires, la stabilité des sous-ensembles, les angles de rotation du cou et flexion du corps. Tous ces critères sont présentés par des coefficients sans dimension calculés automatiquement par l'application développée, permettant ainsi d'évaluer la complexité des séquences de désassemblage. A cet effet, un environnement mixte de réalité virtuelle pour le désassemblage (VRDE) est développé, basé sur le langage de programmation Python, en utilisant deux bibliothèques : VTK (Visualisation Toolkit) et ODE (Open Dynamics Engine), les formats d'échange étant fichiers: STEP, WRL et STL. L'analyse des résultats obtenus démontrent la fiabilité de l'approche proposée fournissant ainsi une aide non négligeable pour l'évaluation des séquences de désassemblage lors de processus de développement de produits (PDP). Les autres conséquences de ce travail consistent à classer les critères en fonction de leur importance. A cet effet, des coefficients de modération peuvent être attribués à chacun d'eux permettant ainsi une méthode d'évaluation plus complète.

Mots-clés: séquences de désassemblage, graphe de désassemblage, analyse géométrique, analyse d'amovibilité, réalité virtuelle.