Software Lab
Computational Engineering Science
Pusher Mechanism

Aaron Albert Floerke, Arseniy Kholod, Xinyang Song, Yanliang Zhu

Supervisor: Dr. rer. nat. Markus Towara*

18th November 2024

*Informatik 12: Software and Tools for Computational Engineering, RWTH Aachen University, info@stce.rwth-aachen.de

Contents

Preface

The topic "Pusher Mechanism" was assigned as a final project by the Department of Informatik 12: Software and Tools for Computational Engineering, RWTH Aachen University, for the Software Lab course in the Computational Engineering Science B.Sc. program. This work was carried out under the supervision of Dr. rer. nat. Markus Towara.

This project involves developing an enhanced version of the well-known planar four-bar linkage, a mechanical system widely used in applications like conveyor systems, oil well pumps, and robotic arms. By adding an extra joint, the extended mechanism offers more degrees of freedom, making it better suited for specific tasks. In this work, we designed and implemented this extended four-bar linkage to find a suitable mechanism for moving a box along a conveyor while avoiding obstacles.

In the first phase of the project, we analyzed the user requirements provided by our supervisor and broke them down into system requirements. This was followed by a theoretical analysis of the mechanism's geometry. Based on this analysis, we selected Python as the implementation environment due to its suitability for the task and the team's expertise.

The implementation consists of three interconnected components. First, the backend was developed to handle the geometry of the linkage, calculating the coordinates of all joints based on input parameters to ensure accurate modeling.

The second component is the frontend, a graphical user interface (GUI) created with the Tkinter library. It enables users to visualize the linkage's movement, modify its parameters, and display essential information about the mechanism.

Additionally, a well-documented testing process was carried out to ensure the correctness and reliability of both the backend and frontend. This testing verified the system's performance across various scenarios, ensuring its accuracy and robustness.

With our implementation, we successfully addressed the optimization problem of moving a box along a conveyor while avoiding obstacles. The addition of an extra joint to the four-bar linkage provided the necessary degrees of freedom, enabling precise trajectory design. This solution met the system and user requirements and demonstrated the mechanism's effectiveness in achieving task-specific motion.

Furthermore, detailed documentation of the software and project management processes was created to enhance maintainability and offer a clear understanding of the project's structure. This documentation ensures that future developers or users can efficiently modify and extend the system. Overall, this work demonstrates the successful combination of theoretical analysis, design, and practical implementation in creating a functional pusher mechanism.

1 Introduction

The industrial revolution in the 18th and 19th centuries brought about significant advancements in manufacturing processes, one of which was the challenge of transporting products efficiently between various workstations in factories. A key solution to this challenge was the development of mechanical systems like the four-bar linkage, which can be used as a pusher mechanism to move products along production lines or between different conveyor systems. Despite its simple structure, the four-bar linkage has proven to be an effective mechanism in various industrial applications.

Over time, the four-bar linkage model has expanded beyond basic conveyor systems and has found applications in more complex systems, such as pumpjacks, robotic arms, and automotive engineering. Its simplicity and efficiency continue to make it relevant in modern mechanical design.

In this work, we aim to analyze the theoretical principles behind the four-bar linkage, design and implement an extended version of the mechanism with an additional joint, and apply it to solve the problem of moving a box along a production line while avoiding obstacles. By enhancing the classic four-bar linkage, we seek to provide a more flexible solution suited to complex real-world tasks.

The structure of this paper is as follows: In Section 2, we analyze the user requirements, derive the system requirements, and provide an overview of the theoretical analysis of the mechanism's geometry. Section 3 covers the selection of the implementation environment, taking into account the system requirements and the team's expertise, as well as the preparation of UML class models for the implementation phase. Section 4 describes the implementation of the four-bar linkage, the graphical user interface, and the software testing process, while Section 5 provides software documentation to ensure its maintainability. In Section 6, we use the developed software to determine the appropriate mechanism parameters to move the box along the conveyor. Finally, in Sections 7, we discuss our project management.

2 Analysis

2.1 User Requirements

user requirements explained (includes essential information and references into literature on technical background of the topic, e.g, [1]) based on UML Use Case diagram(s)

2.2 System Requirements

functional and non-functional system requirements explained

2.3 Theory

explain geometry

3 Design

3.1 Principal Components and Third-Party Software

libraries that you built on explained briefly and references to further information

3.2 Class Models

UML Class diagram(s) and description; should link into overall design through reference of application programming interfaces (API) of third-party software

4 Implementation

4.1 Development Infrastructure

programming language, compiler, run time libraries, target platform (hardware, operating system)

4.2 Source Code

overview of source code structure (file names, directories); build instructions; references into source code documentation e.g, doxygen[1]; short (!) code listings

```
1 #include<iostream>
2 int main() {
3   std::cout << "Leave me alone world!" << std::endl;
4   return 42;
5 }
```

if helpful (must come with detailed explanation)

4.3 Software Tests

e.g, googletest[2]

5 Documentation

6 Optimization Problem

7 Project Management

who did what, when, and why; organization of collaboration, i.e. [online] meetings, software version control (e.g, git[3]

References

[1] Adam Ries. *Rechenung auff der Linihen und Federn*. Annaberg, 1522.

A User Documentation

A.1 Building

e.g, using cmake[4] and make[5]

A.2 Testing

e.g, `make test`

[1] `https://github.com/doxygen/doxygen`
[2] `https://github.com/google/googletest`
[3] `https://git.rwth-aachen.de`
[4] `https://cmake.org/`
[5] `https://www.gnu.org/software/make/`

A.3 Running

documented sample session(s); e.g, `make run`