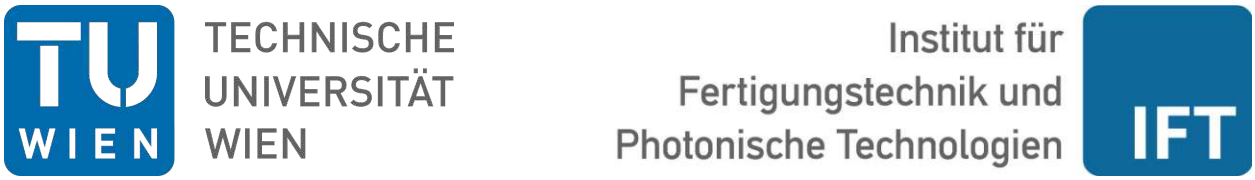

Masterarbeit

Automated Waypoint Extraction for a Cleaning Apparatus for Space Applications

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Master of Science (MSc) unter der Leitung von

Ass.Prof. Dipl.-Ing. Dr.techn. Thomas Trautner
(Institut für Fertigungstechnik und Photonische Technologien)

Dipl.-Ing. Gernot Mauthner
(Institut für Fertigungstechnik und Photonische Technologien)

eingereicht an der Technischen Universität Wien

Fakultät für Maschinenwesen und Betriebswissenschaften

von

BSc Benedikt STINGL

Matr.Nr.: 01326239

Wien, im März 2024

Benedikt Stingl

Ich nehme zur Kenntnis, dass ich zur Drucklegung meiner Arbeit unter der Bezeichnung

Masterarbeit

nur mit Bewilligung der Prüfungskommission berechtigt bin.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass die vorliegende Arbeit nach den anerkannten Grundsätzen für wissenschaftliche Abhandlungen von mir selbstständig erstellt wurde. Alle verwendeten Hilfsmittel, insbesondere die zugrunde gelegte Literatur, sind in dieser Arbeit genannt und aufgelistet. Die aus den Quellen wörtlich entnommenen Stellen, sind als solche kenntlich gemacht.

Das Thema dieser Arbeit wurde von mir bisher weder im In- noch Ausland einer Beurteilerin/einem Beurteiler zur Begutachtung in irgendeiner Form als Prüfungsarbeit vorgelegt. Diese Arbeit stimmt mit der von den Begutachterinnen/Begutachtern beurteilten Arbeit überein.

Ich nehme zur Kenntnis, dass die vorgelegte Arbeit mit geeigneten und dem derzeitigen Stand der Technik entsprechenden Mitteln (Plagiat-Erkennungssoftware) elektronisch-technisch überprüft wird. Dies stellt einerseits sicher, dass bei der Erstellung der vorgelegten Arbeit die hohen Qualitätsvorgaben im Rahmen der geltenden Regeln zur Sicherung guter wissenschaftlicher Praxis „Code of Conduct“ an der TU Wien eingehalten wurden. Zum anderen werden durch einen Abgleich mit anderen studentischen Abschlussarbeiten Verletzungen meines persönlichen Urheberrechts vermieden.

Wien, im März 2024

Benedikt Stingl

Danksagung

Diese Arbeit wäre ohne die Unterstützung von sämtlichen Personen nicht möglich gewesen.
Mein besonderer Dank gilt meinem Betreuer vonseiten der TU Wien, Thomas Trautner, für die durchgehende Unterstützung bei diesem Projekt. Weiters möchte ich dem gesamtem Team der Cleanliness bei OHB für die Unterstützung und die Möglichkeit dieser Arbeit meinen Dank aussprechen. Besonders hervorheben möchte ich dabei Axel Müller, Diana Urich und Volker Böhm die, obwohl der eigentliche Betrieb schon genug Herausforderungen mit sich brachte, immer ein offenes Ohr hatten.

Weiters möchte ich Alexander Kinzer für den Kontakt zum Institut danken, ohne dem diese Arbeit nicht möglich gewesen wäre. Zuletzt möchte ich meinen Eltern für die Unterstützung danken, die ich in den letzten Jahren erhalten habe.

Für Papa.

Kurzfassung

Partikuläre Kontamination kann zu Störungen in optischen Systemen von Satelliten führen und kann in allen Lebensphasen eines Satelliten auftreten, speziell durch Umverteilung beim Start. Das ist einer von vielen Gründen, warum Satelliten in Reinräumen zusammengebaut, integriert und getestet werden. Um pertikuläre Kontamination zu entfernen, müssen die van-der-Waals-Kräfte überwunden werden, die die Partikel auf Oberflächen haften lassen. Das kann durch die Anwendung von CO2-Schneereinigung erreicht werden.

Bisher wurden bei OHB (Orbitale Hochtechnologie Bremen) kleine und weniger komplexe Bauteile per Hand gereinigt. Größere Strukturen, mit komplexeren und engeren Strukturen, können nicht mehr per Hand gereinigt werden, ohne eine Gefahr für Personal oder die Hardware darzustellen. Dies machte einen Reinigungsapparat notwendig, der ein Punkterezept als Input für den Reinigungsvorgang einliest. Der CO2RE (CO2Reinigung) Reinigungsapparat wurde speziell dafür designed. Er hebt und senkt die Hardware über eine Gabel und hat eine Lanze, die sich in der xy-Ebene bewegen kann. Um Schäden durch den Reinigungsapparat zu vermeiden, und die damit einhergehenden Konsequenzen, wurde ein Sicherheitsabstand zwischen der Lanze und der zu reinigenden Hardware definiert.

Das Ziel dieser Diplomarbeit besteht darin, Werkzeuge zu entwickeln, die die Extraktion von Punkten, welche die Position der Lanze darstellen, an denen gereinigt wird, zu automatisieren, die die Bewegung zwischen diesen Punkten beachten und die Validierung des Gesamtprozesses vereinfachen sollen.
Basierend auf dem V-Modell wurde der Reinigungsapparat als gesamtes System analysiert, was zu Zielen und einer Liste von Anforderungen führte. Sie bilden die Basis für das Konzept und Entscheidungen hinsichtlich der Programmierung. Die Implementierung des Algorithmus, durch welchen die Lanze während der Durchführung nicht näher als den Sicherheitsabstand kommt, wird präsentiert. Die Extrahierung der resultierenden Konturen und die Wahl von Punkten entlang dieser Konturen anhand einer vordefinierten, maximalen Länge, ergibt die Positionen für die Reinigung. Um zu garantieren, dass der Sicherheitsabstand zwischen den Reinigungspunkten eingehalten wird, wird der A* Algorithmus auf binären Bildern angewandt. Um die Anzahl an Eckpunkten zu reduzieren, wurden 'Autobahnen' implementiert, welche niedrigere Kosten entlang bestimmter Zeilen und Spalten des Graphen mit sich bringen.
Ein Prozessvorschlag zur Verwendung des Hauptprogramms, in dem die Punktextraktion geschieht, und die Werkzeuge für die Verifikation, wird vorgestellt.

Nach der Validierung des Algorithmus anhand von verschiedenen Geometrien, begründet die Diskussion die Einhaltung der Anforderungen. Zuletzt werden Vorschläge für zukünftige Arbeiten zu der korrekten Platzierung der Konturen, einem höheren Automatisierungsgrad, der Verteilung des CO2-Strahls und der Wegfindung aufgelistet. Der präsentierte Algorithmus kann als Grundlage für zukünftige Entwicklungen in der automatisierten Extraktion von Reinigungspunkten dienen, um CO2-Schnee zur Reinigung von Flughardware zu applizieren.

Abstract

Particulate contamination can lead to disturbances in the optical train of satellites and can redistribute in all phases of a satellite's life time, especially during launch. This is one of the many reasons satellites are assembled, integrated and tested in clean rooms. To remove particulate contamination, the van-der-Waals forces holding particles to surfaces must be overcome. This can be achieved by using CO2 snow cleaning.

Until now, at Orbitale Hochtechnologie Bremen (OHB) only smaller and less complex structures were cleaned by hand. Larger structures with more complex or narrow features cannot be cleaned by hand without risking damage to personnel or hardware. This makes a cleaning apparatus necessary which reads a point recipe as input for cleaning operations. The CO2 Reinigung (CO2RE) mobile apparatus was designed to accomplish this by lifting and lowering hardware with a fork and a lance, containing the CO2 nozzle and a laser distance sensor and which can move in the xy-plane. To avoid damage to hardware caused by the apparatus, as well as all following consequences, a safety distance was determined between the lance and the hardware to be cleaned.

The goal of this thesis was to develop tools to help automate the extraction of points representing positions of the lance for cleaning flight hardware while also considering the movement between such points and to facilitate the validation of the process.
Based on the V-model, the apparatus as a system is analysed resulting in objectives and a list of requirements. They form the basis for the concept and programming choices. The implementation of an algorithm is presented with which the lance does not come closer to hardware than the safety distance at any position during operations. Extracting the resulting contours and selecting points at a maximum predefined distance along the contour serves as a basis for the resulting cleaning positions. To guarantee the safety distance is kept between such cleaning points, the A* algorithm is applied to binary images. To reduce unnecessary turns, 'highways' were implemented which lower the weights in distinct columns and rows of the provided graph. A suggested process to use the main program for the point extraction and the tools to verify the distance from the part of interest are also presented.

After the validation of the algorithm to various geometries, a short discussion argues to what extent the requirements are met. Finally, suggestions for future work on the correct placement of contours, a higher level of automation, coverage of the CO2 jet and pathfinding are given. The presented algorithm can serve as a basis for future development to be able to automatically extract cleaning points for the application of CO2 snow to flight hardware.

Acronyms

acp	advanced clean production
ANN	Artificial Neural Network
CAD	Computer Aided Design
CO2RE	CO2 Reinigung
OHB	Orbitale Hochtechnologie Bremen
PAC	Percent Area Covered
SCF	Supercritical Fluid

Contents

1. Introduction

Cleanliness or contamination control, which is „any organized action to control the level of contamination“ [1], has become influential in many different technical fields. While being in use for decades in the context of space flight [2], it is an important part in the manufacturing of microelectronics or in the pharmaceutical industry. In recent years the automotive sector started to recognize the advantages contamination control brings with it, e.g. performance increase [3]. Such aspects are also relevant in the context of spacecraft where gains in performance can, for example, mean a maximum of (scientific) data, or, smaller systems leading to lower mass and lower launch costs. This can be reached by e.g. the usage of clean rooms, organizational measures and cleaning methods.
Generally speaking, what contaminants are depends on the process or hardware that is meant to be protected. Of main concern in most industries are particulate and molecular contamination [2, 4] where particulate contamination will be the main focus in this document. Depending on the context, biological contamination can also be of concern, especially for machines and equipment that aim to find signs of life [2] or in the context of crewed space flight.
At OHB the cleanliness team is responsible for achieving and maintaining the cleanliness levels required for different satellite missions. Especially optical systems, which are extremely sensitive to contamination in general [4], are one focus of the team. As will be further elaborated in this document, not only the optical systems themselves but all surfaces which are not physically separated from these sensitive systems have to fulfill stringent levels of cleanliness to ensure the best possible performance in orbit. While smaller structures from an earlier integration state or simple geometries can be cleaned by hand, larger structures, especially their inside surfaces, can lead to long and exhausting cleaning sessions, causing potentially inconsistent cleanliness results and increased risk to flight hardware through human error. Thus, the need for a cleanliness apparatus was found which can be programmed to give more reproducable results, reduce the risk of damage and better documentation.
This thesis focuses mainly on the cleanliness aspect of particles while also providing context to cleanliness in general and a focus on cleaning with supercritical CO2. An overview of the application of CO2 cleaning at OHB is given following an introduction to the CO2RE mobile apparatus. The current status of relevant topics to this thesis is also given such as pathfinding in the industry.
Next, it is shown how and to what extent the process of extracting points for cleaning the inside surfaces of larger structures can be automated and which tools can be used to validate them. A recommended process is given showing which steps or verifications follow which in the process. Next, the results of this extraction process to different geometries are shown. Finally, recommendations for future development is given.

2. Current State of Technology

2.1 Particulate Contamination

Particulate contamination is concerned with particles which are airborne or on surfaces [1]. Particles themselves are defined by [1] as a "unit of matter with observable length, width and thickness". In everyday life, what is known as dust can also be considered as an example for particulate contamination [4].
Particles can be differentiated e.g. in terms of their size, form, chemical composition or their optical and electrical properties. The most important property is their size since it has the largest influence on their behaviour when being airborne [2, 5]. This mixture of gas and particles is referred to as aerosols. Particles in aerosols are subject to different influences such as gas molecules and (field) forces. Depending on the size of a particle, different physical phenomena, e.g. gravity, can form the main contribution to its behaviour [2]. To describe the phenomena, an equivalent diameter is used which shows the same behaviour as the particle of interest does in the considered process. This means that equivalent diameters can be defined with respect to e.g. the size and shape of a particle, its aerodynamic, optical or electric behaviour [2, 5].
Sources of particles in aerosols can be all sorts of processes and mechanisms but humans are the main contributors [2] leading to the fact that most of the contamination of a satellite can be derived from ground operations [4]. Particles can form from the gas phase through nucleation by sticking to a nucleus particle, but also through coagulation, where two particles interact with each other [2, 5]. As particles move through the air, they can come close to surfaces where van-der-Waals forces can overcome other acting forces which leads to the particle sticking to the surface [2], possibly converting it to a contaminant when sticking to the right surfaces. The van-der-Waals forces holding the particle to the surface are dependant e.g. on the size, the composition or the roughness of the particle but also on properties of the surrounding air, such as temperature and humidity [2].
Particulate contamination influences different components of a satellite e.g. through its size by blocking off light and through scattering light but can also lead to higher wear on mechanisms and cause short circuits. The area on a surface blocking off light is called Percent Area Covered (PAC), which is an important measure in determining the cleanliness of satellite components [4]. Consequently, the amount of light which enters an optical system is reduced by PAC. The effect increases the further the particles lie along the optical path with the worst case of an electrically conducting particle short circuiting several pixels in the focal plane [4]. To avoid any sunlight from entering the optical train of a satellite, baffles are designed to block solar light off the insides. Other methods, such as guiding the incoming light onto as many absorptive surfaces as possible, are also incorporated to minimize stray light reaching the sensitive sensors of a satellite [4]. Such absorptive surfaces are more effective the cleaner they are.
Contaminants can migrate to other surfaces inside a closed volume during any point in a satellite's lifetime, including testing and preparation but especially during launch [4]. In orbit, particles can also come off and redistribute through processes such as opening sensor covers or thermal contractions. Thus, it is important to remove contaminants from any surface of a volume of interest if it contains sensitive equipment which is not shielded by physical barriers. Collisions with orbital debris can also generate particulate contamination [4].

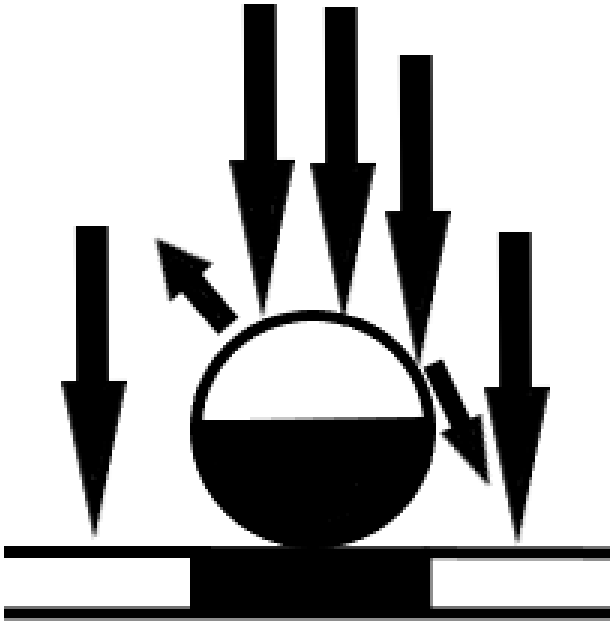

Abb. 2.1: Illustration of PAC and how a particle blocks light from entering e.g. an optical lens as well as scattering caused by its presence, based on [4]. This can cause sever loss in the quality of collected data.

Optical systems are not the only components suffering from particulate contamination. Thermal radiators and solar cells also suffer from performance degradation. In the case of thermal systems, a particle on a radiator leads to absorption of solar radiation from the sun instead of reflection. When an equilibrium temperature is reached, it lies higher than without particular contamination. In fact, the effect on emissivity, temperature and equilibrium temperature scales linearly with PAC [4]. While the effect of PAC to thermal radiators scales linearly, the effect of PAC to solar cells is nonlinear for the case that all cells receive equal particulate contamination [4]. In fact, scattering is the bigger issue and the relationship between power efficiency loss and PAC can be considered invariant. The case of just one solar cell receiving all contamination leads to the cell becoming a resistor in addition to converting less electrical power [4].

2.2 Clean Rooms

Clean rooms are an important measure in ensuring that the required quality of a product in general can be achieved. By controlling environmental variables such as temperature, humidity, electrostatic discharge, particle concentration and size of particles, the quality of a product can be altered or certain functions made possible in the first place [2]. Specifically particulate contamination is minimized through e.g. the filtering of air, special garment, gowning procedures and organisational efforts. For example, transport of equipment from a lower to a higher class of clean room requires cleaning to avoid contamination. The variables which are needed to achieve the required level of quality can vary depending on the context. The more variables are controlled and the lower the tolerance is the higher are the organisational effort and also the related costs. In the context of particulate contamination DIN EN ISO 14644-1 is used to sepa-

Abb. 2.2: View into the ISO 5 clean room at OHB. Note the holes in the floor to enable downward facing laminar flow. Also note the worn garment and the granite block for optical measurements to minimise contamination and changes in temperature.

rate clean rooms into different classes. The variables defining the classes are the size and respective concentrations of particles. The lower the class (e.g. ISO 5 in comparison to ISO 8) the fewer and smaller particles are allowed. Lower classes of clean rooms bring with them a higher organisational effort which manifests itself in different clothing or more advanced facilities to obtain the necessary certification.
One important aspect in the context of cleanliness is the circulation and filtration of air. [2] shows different ways and possible pitfalls of air circulation but the important point of it is to protect hardware and to transport particles away from hardware. In the ISO 5 clean room at OHB, see Figure 2.2, which has a downward facing laminar air stream, the particles are guided by the air stream through holes into the lower floor where they are guided into filtering units.

The usage of clean rooms or, more general, contamination control also has an economic effect on projects. When done correctly and, more importantly, when incorporated from the beginning of a project, it can lower the costs of quality inspections or, as in the pharmaceutical sector, lower the amount of rejects [2, 4]. [2] lists as reasons too much contamination, bad performance or even destruction which can lead to considering hardware as reject. It has to be kept in mind that quality controls themselves can lead to changes or contamination of the hardware under inspection [2]. Also, inspections can almost never analyze the entire hardware and occur late in process chains, leading to the fact that the causes of contamination or destruction cannot be determined exactly or through destructive measures only [2].
It can be summarized that clean rooms and supporting measures are an essential part of contamination control to ensure the necessary protection of hardware and personnel to achieve the required functionality of hardware.

2.3 Measuring Particulate Contamination

As discussed in previous sections, the number and size of particles, called the concentrations, is an important measure in cleanliness [2]. For example, the classification of clean rooms is determined by the number and size of particles in a defined volume of air. Depending on the application, certain sizes of particles are more relevant than in others. But as mentioned above, different diameters can be defined to describe the behaviour of particles. One physical phenomenon, which is often used to determine the size and number of particles, e.g. to classify and certify clean rooms, is their effect on scattering monochromatic light [2]. Under the assumption that only one particle is measured at the time, a correlation between the particle size and amount of stray light is used to estimate the size of a particle. Thus, it is possible that multiple particles can lead on average to a larger average diameter and lower particle concentrations than when they were measured separately [2]. Obtaining the actual shape and size of particles is possible through the use of so-called witness samples which can be analyzed in specialized laboratories. They can be used to determine the impact of a process on nearby situated hardware throughout assembly, integration and testing.

2.4 Cleaning with Supercritical CO2

Since the first half of the 19th century the existence of the supercritical state is known [6]. In the second half of that century it was found that gases in this state of matter, Supercritical Fluid (SCF) can serve as solvents [7]. Due to the ban of chlorofluorocarbon solvents by the Montreal Protocol, SCFs gained attention in replacing such chemicals [8].

Fig 2.3 shows a schematic phase diagram which is typical for CO2. It shows the regions where the thermodynamically stable phases exist in dependence of temperature (x-axis) and pressure (y-axis). The different phases are separated from each other through the phase boundaries where the neighboring phases exist in equilibrium. The supercritical state can be found beyond the critical temperature and the critical pressure. Far enough from the critical point, away from the supercritical state, gases behave ideally. Approaching the critical point, increasing pressure and/or temperature, the behaviour diverges more and more and can eventually be completely distinct from the often assumed ideal gas behaviour, before reaching the supercritical state which is neither gas nor liquid. The change in behaviour and the distinct properties, lying between thos of gases and liquids, are the reason for why the material is then referred to as fluid. For example, the density typically lies between the ones of its gaseous and liquid forms while the viscosity is similar to the gaseous state [7, 9].
The solvent properties of SCFs can be improved by increasing the density of the fluid which in turn can be influenced via temperature and pressure. While the influence of temperature is more complex, the influence of pressure is much more pronounced [7]. Since CO2 is non-polar, non-polar particles are soluble in SCFs. Especially organic contaminants (hydrocarbons such as lubrication oils) are well dissoluble [7, 9]. To be able to transport polar molecules, supercritical CO2 is enriched with cosolvents [7]. [7, 9] argue that supercritical CO2 is the best compromise to be applied in engineering as a solvent due to its dissolution properties, near ambient critical temperature and

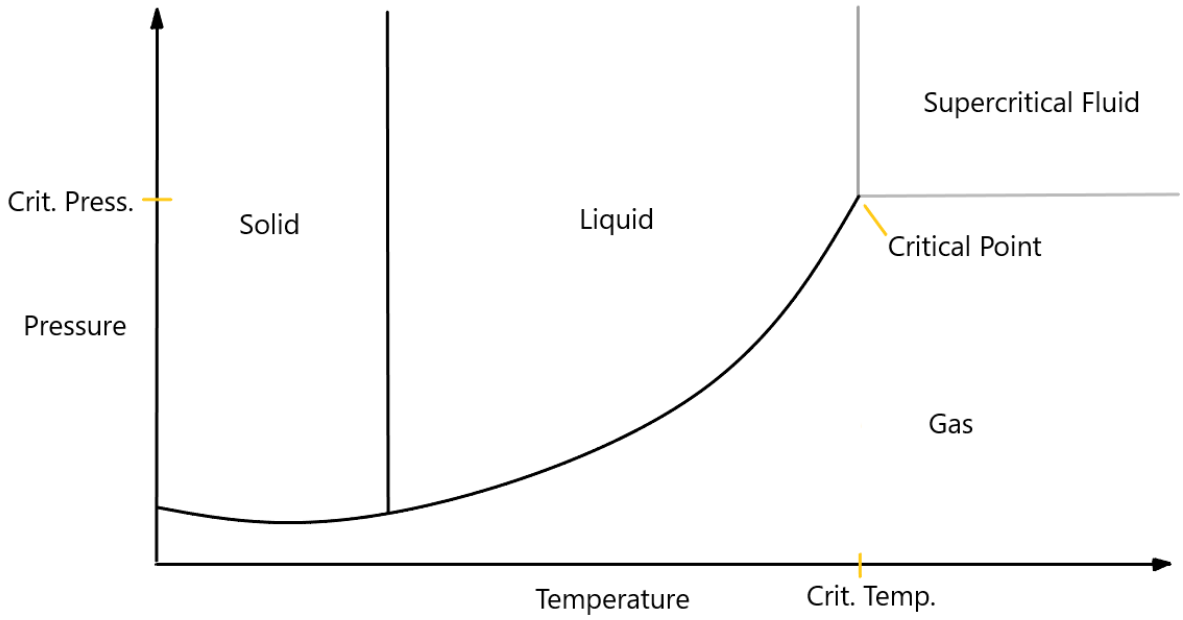

Abb. 2.3: Scematic pressure-temperature phase diagram of CO2, based on [7]. Note that the supercritical fluid lies beyond the critical point (304.1 K and 73,8 bar for CO2 [9]).

relatively low critical pressure, non-toxicity and low cost. Other solvents such as water, ammonia and nitrous oxides have serious disadvantages, often associated with safety. Even though CO2 can displace oxygen in the air, it can be managed by the use of CO2 detectors, proper air circulation and the fact that it is heavier than air and thus accumulates towards the floor. Also, there is no interaction of supercritical CO2 with most common materials with the exception of polymers which can potentially absorb the solvent [7].

As mentioned above, particles can stick to a surface and be held to it by the van-der-Waals forces. To be able to clean a component, it is thus necessary to overcome these forces, and get the particles airborne. Once the particle is in the air, an airflow can carry the particle away from the surface. To achieve this the above mentioned properties of supercritical CO2 are used. As [7] mentions, variables, such as the type and structure of a surface, as well as the contaminant itself and the method of cleaning in general, can influence the achievable level of cleanliness. Specifically for the application of supercritical CO2, the different variables such as distance from the surface of interest, pressure, temperature, surface properties, contaminant and cosolvent can be tweaked to give the optimal results in terms of cleanliness [7].

The process of removing a particle off a surface and making it airborne again is described in detail in [5]. A nozzle which ejects CO2 at the right conditions to form snow, surrounded by a jet stream of clean air is described. Letting CO2 expand through a nozzle is ideally isenthalp, at constant enthalpy. Looking at the pressure-enthalpy diagram in Figure 2.4 this leads to a change in the pressure only and thus a vertical line from the initial state. Depending on e.g. the phase of the feeding source of CO2, its pressure and temperature, the amount of snow can be influenced [5]. Yet, at the end of the cleaning process, gaseous CO2 is always the end product, independent of the initial state of the CO2 [5].

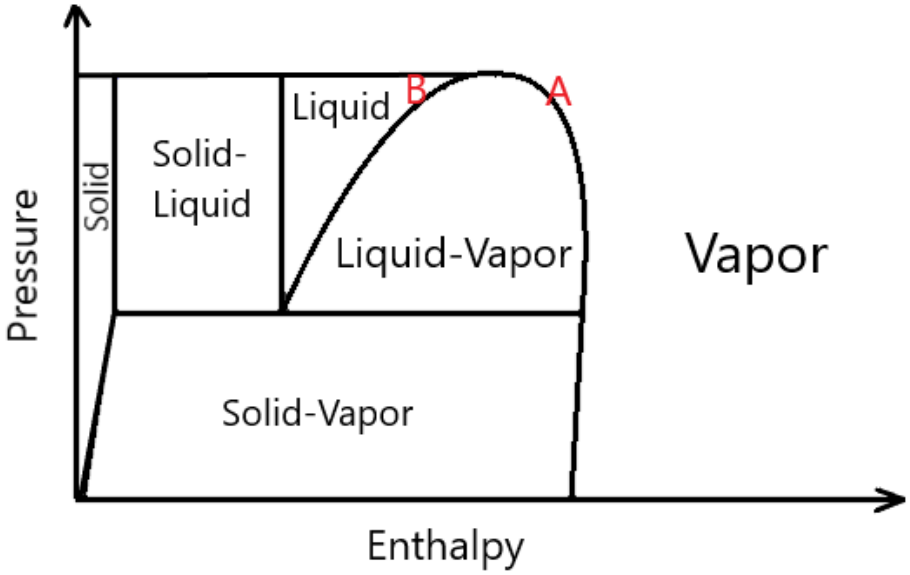

Abb. 2.4: Schematic Enthalpy-Pressure diagram for CO2, based on [5]. With this diagram the different phase changes when exiting an orifice can be explained. It is, ideally, an isenthalp process which can be drawn with a vertical line in the diagram. [5] further mentions that starting from point B in the diagram gives a higher amount of CO2 snow than when starting from A.

As described by [5], when the snow hits the surface with e.g. particulate contaminants, several phenomena take place. First of all, momentum transfer of the snow can overcome the van-der-Waals forces and lift some particles from the surface. Aerodynamic forces themselves play a small role in making particles airborne due to the forming boundary layer. Yet, the CO2 snow which reaches the surface can collide with it. Assuming the snow crystal is of ideal spherical shape, looking at Figure 2.5 the process of collision is illustrated. Upon impact, deformations caused by the impact lead to local high pressures, leading to the conditions which allow for liquid or, under the right conditions, even supercritical CO2 which acts as a solvent for particles, removing it off the surface [5]. When bouncing off the surface, recrystallisation occurs, covering the particles in CO2 snow. The clean air stream surrounding the CO2 stream carries the now enlarged particle away, leaving a cleaner surface behind. Finally, the airflow of the cleanroom further carries the particle away until it finds its way into a particle counter, out of the cleanroom or into a filter unit. Further effects also help in removing particulate contamination from a surface [5]: the steep temperature gradient caused by the CO2 snow introduces local stresses which can break up particles and also lead to a change in Brown'ian motion and thus a force directed towards the lower temperature, also helping in overcoming the van-der-Waals forces. The sublimation of the CO2 snow which lets it expand by a factor of 600 [10] causing bursts near the surface also help removing the contaminants off the surface. It has to be mentioned that the process leads to a separation of charges which is why the nozzle and the cleaned part have to be grounded.

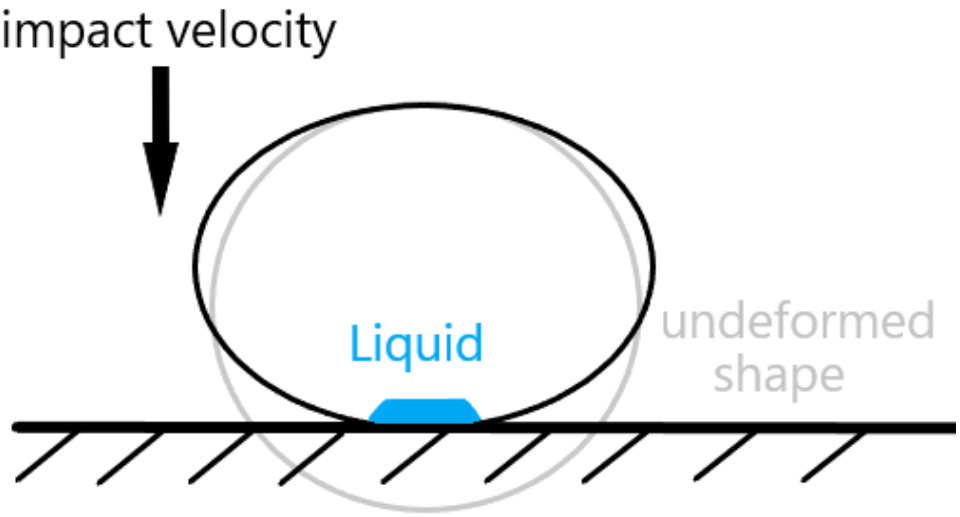

Abb. 2.5: Schematic illustration of the process occurring when an ideally spherical CO2 snow particle hits the surface, based on [5]. The deformations of the snow particle can cause a locally liquid or even supercritical phase. With the presence of a particulate contaminant it is incorporated into the phase and upon bouncing off and recrystallization the contaminant is removed.

Mounting a CO2 cleaning nozzle to a robot to clean spacecraft parts is not a new idea. [11] describes in his paper how a robot mounted on linear rails and with a tilting CO2 ice nozzle cleaned various hardware pieces from the Titan IV rocket and the Space Shuttle. The paper mentions that a teaching mode and a controller to steer the robot are available, as well as a graphical user interface program to perform offline programming of its motions and collision control. However, due to the setup of the robot only flat panels or the outside of larger structures could be covered.
Two more recent examples show the primary mirror of the European Space Agency's satellite Henschel being cleaned using CO2 ice [12]. Also, tests were performed for cleaning the mirrors of the James Webb Space Telescope using CO2 ice, although both examples were cleaned by hand [13].

2.5 Pathfinding

Pathfinding must run fast, on low resources and also give the best path under the given constraints. Applications vary from video games over robotics, agricultural and maritime applications to crowd simulation. Searches can be differentiated into many categories such as single agent or multi agent, static, dynamic or real-time environments and informed or uninformed [14, 15].
The problem of pathfinding can be split into the graph generation and the pathfinding algorithm [14]. The graph, a general set of points with connections between them, can be constructed using a regular grid. This is a widespread method, e.g. in operating mars rovers [14]. In 2D the discretisation can be achieved using triangles, squares or hexagons. [14] describes how algorithms applied to hexagonal grids outperformed algorithms operating on square grids by having a lower search time and required memory.
A* is a popular informed pathfinding algorithm since it outperforms other algorithms

in terms of flexibility and efficiency with respect to time and memory [15]. [15] mentions that A* can lead to an increase in resource efficiency such as memory or time, of over 40%. The algorithm is based on the heuristic function which combines the cost of movement from the start node to the current node as well as the cost from the goal node to the current node. Depending on the application, different heuristic functions are used to decide on the shortest path which can make the difference in finding the optimum path. This is where A* is flexible and can be adapted.

Other methods of pathfinding include Dijkstra's algorithm, D*, Rapidly-exploring Random Trees and, in more recent years, Artificial Neural Network (ANN)s. While ANNs lack the necessary precision to obtain feasible paths, they can be applied as a first approximation before optimizing using different algorithms [16].

Today, research concerned with A* is focused on multi-agent pathfinding and avoiding collisions between them. Another topic in toadys research deals with the question of which information is actually useful to find the optimum path. Also, each algorithm performs different in varying scenarios such as available resources. As described in [15], informed algorithms perform better when the domain is well known.
Hierarchical Pathfinding A*, a variation of A*, often found the optimum path in less time and seems to be one of the many promising candidates for future applications [15].

As mentioned in the introduction, cleanliness is a topic not exclusive to space flight. It is thus not surprising that several applications exist which use a combination of robotic arms and CO2 cleaning. As promoted by advanced clean production (acp), the automotive sector is using CO2 snow cleaning on car body parts prior to applying coatings [17]. According to [18], acp relies heavily on the application of Artiminds RPS [19] to generate movement patterns for automated CO2 cleaning. They offer a simulation environment using Computer Aided Design (CAD) data to visualize the process. The software requires points in space which have to be set manually where the machine applies the CO2 jet. Yet, the motions of the machine itself are generated automatically where different options can be chosen from. In contrast, the algorithm shown in this document treats the automatic extraction of points and pathfinding between them.

In modern manufacturing, computer numerically controlled machines, such as milling machines or 3D printers, are the current standard. For example, milling machines Cutter Location data files are created which are then converted by a postprocessor to machine specific code. 3D printers in contrast use slicers, which return G-code for steering. Feature detection is still a topic of research with various methods still under investigation such as the application of ANNs [20, 21].

3. Development Methodology and System Design

The following chapters are based on the V-model as described in VDI 2206 [22]. The V-model is a model for product development in the areas of software and mechatronic systems and highlights the interfaces between the different disciplines.
It starts by assessing the initial situation and describing the overall system from which the problem statement, objectives and requirements are derived, which serve as the basis for the architecture and strategy. Following, the implementation describes which algorithm is used and which limitations it brings with it. Finally, the validation terminates the process and a discussion reflects on the fulfillment of the requirements and possible improvements.

3.1 Initial Situation

At OHB the CO2 cleaning is currently done by hand exclusively. An operator uses a 'pistol', see Figure 3.1, or a 'lance' which have a nozzle that creates the CO2 snow encapsulated by a supersonic clean air stream which guides, focuses and accelerates the CO2 snow. The technology is developed by acp who refer to it as the QuattroClean™ technology [10], referring to the effects which are responsible for the cleaning efficiency. When cleaning hardware, the jet is led by hand over all exposed surfaces for multiple runs in a defined manner. When the concentration of particles does not go down any further in comparison to previous runs, the hardware is considered to be clean. Ideally, the concentration decays in an exponential manner. An inspection of the hardware as a whole using white light and UV-light is performed to spot any overseen areas and to confirm the cleanliness status. This process has been applied to a large number of objects and is thus a space qualified standard at OHB.

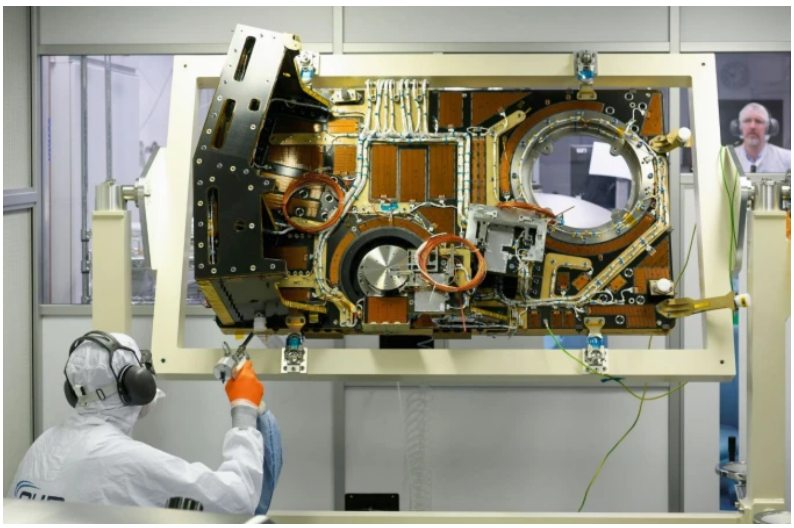

Abb. 3.1: CO2 cleaning done by hand at OHB. The operator sweeps over the part in a defined manner over various runs until no change in particulate concentration is measured.

Abb. 3.2: Available degrees of freedom of the cleaning apparatus. The machine's cartesian coordinate system sits centered on the base plate with respect to x (red) and y (green). The origin of the z coordinate sits on top of the camera of the lance and thus not as depicted here. The coloring is the same as the coordinate system used by Open3D. The rotational degrees of freedom of the lance are the C (black) and R (orange) angles. Also visible in this image are the smaller lance and the laser pointers in the lance head which help orienting the CO2 jet.

Nowadays, larger structures, such as baffles, which can contain optical systems, have to be cleaned, which can also contain semi-closed cavities, which are not accessible by hand. Also, the larger surfaces lead to larger times needed for the cleaning process, which can tire personnel and lead to an increased risk of damage to the hardware. Thus the CO2RE mobile apparatus, also called cleaning apparatus, see Figure 3.2 and 3.3, was designed and built in collaboration with acp to be able to clean larger hardware and to reach into narrow and deep structures, which would otherwise not be reachable by hand. The machine has several operation modes. It is, for example, possible to steer the machine manually via a handheld device. This handheld device also shows the current position of the lance with respect to the machine's coordinate system and allows the activation of the CO2 jet. Another mode is the automatic mode, which reads a point recipe in csv format, which contains all coordinates and angles. Table 3.1 shows an example of such a point recipe. After steering to the origin of the machine's coordinate system, the point recipe is followed from top to bottom until the end of the recipe is reached. The activation and deactivation of the CO2 jet is, however, in the current state of the machine not automated. The automation of the cleaning process lowers the risk of damage to the hardware and helps in obtaining more reproducable particle concentration results since a clearly defined program is executed.

The cleaning apparatus itself consists of a fork which can lift or lower hardware and a lance which can move in a xy-plane. The nozzle in the head of the lance uses the above presented technology by acp for the CO2 cleaning together with a JetWorker which provides the right conditions to create the CO2 snow. The CO2 is supplied by gas bottles which contain liquid CO2 under 300 bar pressure at 20 °C. A compressor delivers filtered air which is used to encapsulate the CO2 stream and to provide the

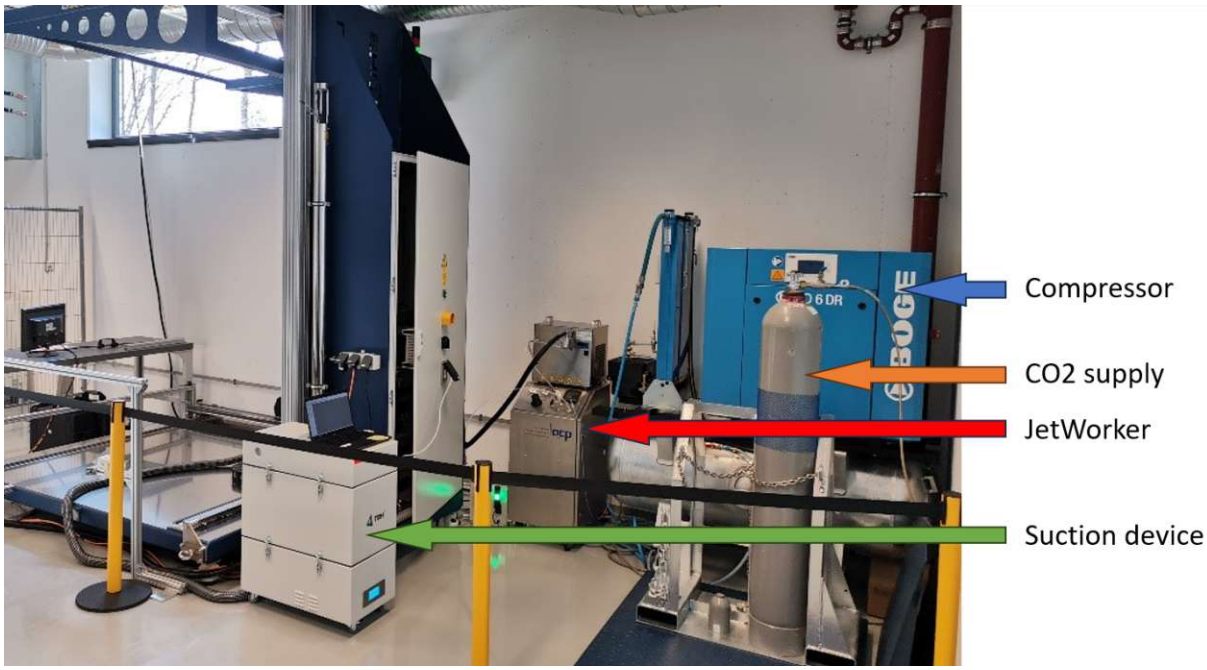

Abb. 3.3: Devices needed for cleaning operations. To operate the machine without cleaning only the compressor is needed which provides the necessary air pressure to lift the fork. Depending on the mass put onto the machine the pressure must be set accordingly.

Tab. 3.1: Example of a point recipe. It contains for each point the position (x, y, z) in the machine's coordinate system as well as the C and R angles. If there is no difference in the angles, e.g. $C_1 = C_2$, see the second point set, the lance is not rotated. This is used to insert corner points which help guiding the lance without starting the cleaning procedure at that point.

X (mm)	Y (mm)	Z(mm)	R1 (deg)	R2 (deg)	C1 (deg)	C2 (deg)
-49	-48	50	75	15	-14.41	-74.41
29	-48	50	0	0	0	0
29	-49	50	75	15	-90.62	-150.62

necessary pressure to lift the hardware on the fork. Upon application of the CO2 jet, the aerosol is sucked through the lance itself through the use of a suction device, which removes the particles that were lifted from the surfaces.
The lance head can change the elevation of the nozzle, referred to as the R axis. Rotation normal to the xy-plane is achieved by rotating the lance as a whole (C axis). Figure 3.2 shows the various available degrees of freedom of the machine. It has to be noted that the machine can only move along one axis at a time, where between two points the machine moves first along the z coordinate, which is followed by the x coordinate and then the y coordinate. The lance head rotates in the direction the lance is moving to allow the laser distance sensor mounted in the lance to detect possible collisions. The lance cannot rotate further than $\pm 180°$ in the C coordinate. If the C1 and C2 and the R1 and R2 angles differ at a given point a so-called CR run is done which goes through the various angles to distribute the CO2 snow. Such points are referred to in this document as cleaning points. If no CR run is performed, the points are referred to as corner points.

3.2 Problem statement

Until now, the process of creating a point recipe was done mostly by hand in a long and error-prone process using offsets from surfaces from a 3D CAD file. Some macros exist which cover basic geometrical cases, such as cylinders. Due to organizational changes in the company, the access to CAD software became more difficult to the team operating the cleaning apparatus.
Following the creation of a point recipe, a dedicated software was used to determine if collisions would occur, referred to as virtual run. It has to be mentioned that collisions in the context of the software are situations where the lance is positioned such that the safety distance is not kept anymore. This is essential, since actual collisions carry with them a long list of actions which are required to ensure that the performance and reliability of a satellite operating many years in orbit is not influenced by the incident. The software used to detect collisions had several issues:

- It is not clear, where the hardware or the lance are located from the software alone and which changes have to be made to correct for the difference between the part's and the machine's coordinate system.
- Only the position of the jet nozzle is considered for collision control, not the entire length of the lance.
- There is no visualisation of the distances.
- It is rather slow, taking multiple hours for hardware the size of baffles.
- It does not use the exact coordinate positions but rounds to the next mm, and it only returns the shortest distance rounded to decimal mm.

The first point makes it tedious to work out the relative positions inside the software. Coupled with the slow execution, it takes time to understand what is happening and which changes have to be made. The last point in the list is important to note since the handheld device of the cleaning apparatus returns the position in x, y and z up to two decimals of a millimeter. During operations, when the software is run in parallel to show a digital twin, collisions can occur where the virtual run did not detect any.

Finally, pathfinding between the extracted cleaning points was never adressed. Errors in pathing were detected using the virtual runs and points for steering were inserted manually into the point recipe before rerunning the virtual run. This of course prolonged the already long validation process.

3.3 Objectives

The aim of this thesis is the development of tools which can be used to automate the extraction of cleaning points, which represent the center of the lance at the height of the CO2 nozzle. Another objective is to develop a concept to solve the pathfinding between the extracted cleaning points, and to create tools for the validation of all points/positions of the lance.

3.4 Requirements

The following requirements can be derived from the given problem and the defined objectives:

REQ1: At all positions occupied by the lance the minimum safety distance of 100 mm from the center of the lance must be kept along the whole length of the lance.

REQ2: The specialities regarding the behaviour of the machine (movement only in one coordinate, order of coordinates, limitation of the C axis) must be accounted for.

REQ3: The algorithm must take as input a common 3D CAD file format.

REQ4: The output format of the algorithm must be a csv file of the correct format containing all necessary values for automatic operation.

REQ5: In a practical application of a point recipe no collision and no interruption by the laser distance sensor must occur.

REQ6: The maximum distance of cleaning points shall not exceed 120 mm.

REQ7: The algorithm shall not rely on specialized CAD software.

REQ8: The time necessary to validate a possible point recipe shall be reduced and a visualization of the distances shall be available.

REQ9: The algorithm shall be applicable for inside or outside cleaning and shall apply the CO2 jet to the surfaces of interest such that they are covered.

The requirements will be referred to as e.g. REQ1 throughout the document.

3.5 Research Question

The research question of this thesis can be formulated as: How and with which limitations can, given the requirements, the points for cleaning of flight hardware and steering of the cleaning apparatus be extracted in an automatic manner?

3.6 Concept, Software and Architecture

As discussed in previous chapters, modern machining processes are concerned with e.g. feature decomposition, tooling choices and an efficient sequence of machine operations. Due to the unique way the cleaning apparatus is designed, many issues in manufacturing are not of concern in this context. For example, no tooling choice has to be made and there is a tolerance interval of 20 mm (see REQ6) in which the CO2 jet remains efficient, which is orders of magnitude larger than usual in manufacturing processes. Also, feature detection or the exact choice of material is not of concern. Instead, an algorithm using a similar strategy to voxelization was chosen which is described in detail in this chapter.

Concerning the choice of programming languages, Python was chosen due to several reasons:

- It is a well readable language overall.
- There are a wide range of libraries available which can perform well in terms of resources and with regards to performance.

The choice of libraries and the specific version of Python need a deeper look. The thesis can be split into the following major topics: point extraction, pathfinding, collision control and visualisation.
For pathfinding, the A* algorithm in the python Pathfinding library [23] was chosen. It uses arrays/matrices with their entries acting as weights as a graph. Entries in the range of $(-\infty, 0]$ represent obstacles, weights in the range of $(0, \infty)$ represent weights of accessible positions and represent the cost to access them. From this concept the idea arose to use binary images, which can be represented by a matrix, as indicators for forbidden or accessible positions of the lance. Images also lead to the advantage that powerful libraries, such as OpenCV [24], with optimized and tested functionalities are applicable. Finally, the Pathfinding library can be set to reflect the movement of the cleaning apparatus by excluding diagonal movement (REQ2).
Open3D [25] was chosen as an interface between 3D CAD data, specifically stl (REQ3, REQ7), and Python as well as an optimized library to extract and visualize information from such files. Open3D uses OpenMP, a multiprocessing library, to parallelize its execution and to make use of all cores of a computer.
As an example, initially the strategy was to slice structures at defined heights, obtain the contours, extract the corner points and create images using matplotlib. For a specific structure this process took roughly 30 minutes and was not robust. In comparison, using Open3D, the same process took around 3 minutes and was much more robust. This also shows how libraries which are created, tested and maintained by a community are more powerful than self-developed libraries.
With the libraries chosen to be applied to solve the point extraction and the pathfinding, the interface between them are matrices describing possible positions for the lance. As a result, Open3D is used to extract the distances at different heights, giving the pathfinding possible positions to steer between cleaning positions. OpenCV is used to detect the boundaries/contours between forbidden and possible positions of the lance, giving potential positions for cleaning activities. With Open3D, the main limitation was that Python 3.10 had to be used, despite higher versions being available at the time.
For the manipulation of matrices, Numpy was used where applicable. Due to the re-

quirement of a csv file for the point recipe (REQ4), pandas was used to read, manipulate and write tabular data.
To visualise the distances between the part of interest and the lance hvplot [26] was chosen since it can read pandas DataFrames and returns interactive plots by using Bokeh in the background. The disadvantage of hvplot is that it could only be applied using Jupyter notebooks.

As a result of the above considerations, the three-tier architecture can be considered where the above tier is dependant on the lower one [27]. Figure 3.4 shows which libraries are used in which tier. The main work of this thesis is focused on the logic-tier, where information provided by the back-end is processed.

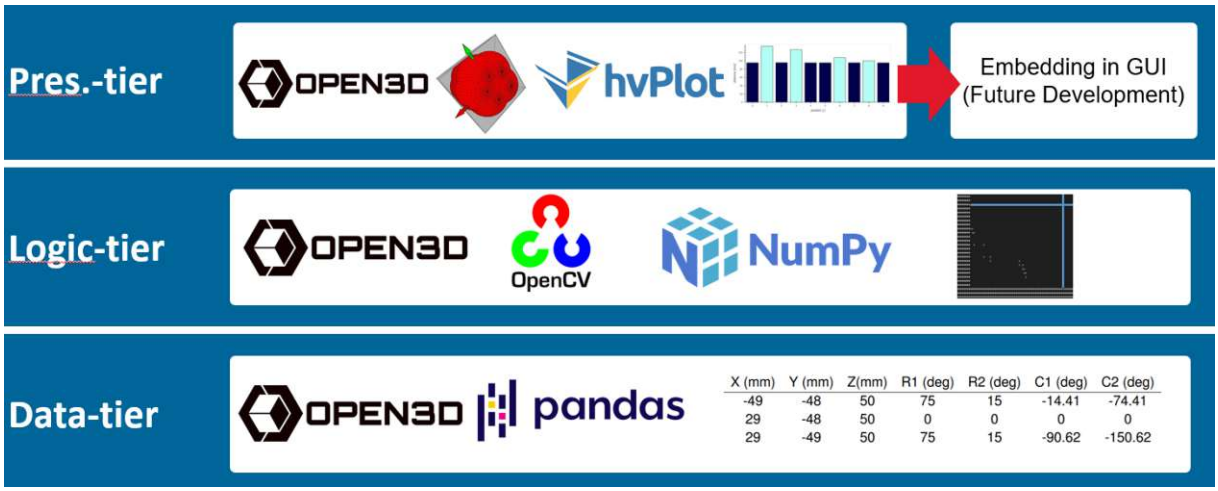

Abb. 3.4: Programming architecture of the thesis, based on the three-tier architecture. The different libraries in the different tiers are shown, as well as some examples on the respective tiers.

4. Implementation

The main program contains all of the necessary functionalities, the cleaning point extraction and the path finding between them, to obtain and view a point recipe. Only specialised tools for visualisations and validation are outside of the main program. The overview over the program is given in Algorithm 1 in the form of pseudocode. Note that Line 18 in the shown algorithm is in fact within its own for-loop for better code readability.

Algorithm 1 Main Program

```
1:  Load part of interest.
2:  Create gridpoints.
3:  if create_new_images then
4:      for height = 0, 1, ..., n_h − 1 do
5:          Create binary images with safety distance.
6:      end for
7:  end if
8:  if blend_images then
9:      for height = 0, 1, ..., n_h − 1 do
10:         Blend images from below.
11:     end for
12: end if
13: Delete custom height information.
14: if extract_new_points then
15:     for height = 0, 1, ..., n_h − 1 do
16:         for contour = 0, 1, ..., n_c − 1 do
17:             Select, extract and save relevant contour points.
18:             Determine reference point.
19:         end for
20:     end for
21:     Apply polar centre macro to all points.
22: end if
23: if create_paths then
24:     Create corner points and whole path.
25: end if
26: if visualise_point_recipe then
27:     Visualize point recipe.
28: end if
```

4.1 Settings and Structure

The settings in the main program are sorted into

- numerical
- strings
- binary/switches

Numerical settings are e.g. the safety offset or part translation in x, y and z. Strings are mainly used for folder names but are also used to differentiate between part or machine bounds. Switches are used to steer the algorithm at if-statements as shown in Algorithm 1. Due to the complexity of baffles it is possible that the process of point extraction requires confirmation or interaction by the user at different times. In case a previous milestone was achieved, e.g. in case the image creation, see below, is already completed and the algorithm is rerun, this step can be skipped using the switches. Finally, some switches are used to manipulate the visualisation, e.g., showing the mesh of the part of interest. A complete list of all settings with comments is provided in Table 4.1.

4.2 Point extraction

The goal of the point extraction is to obtain a point recipe of all points where cleaning is supposed to be done. It is part of the main program and can be found in Algorithm 1. In contrast, the points for path control are added in a later stage and are shown in a separate algorithm.

Starting the point extraction, the first action is to load the part of interest. A function takes the loading path of the stl-file (REQ3) as well as the translations and applies them to the part. The translations correct the position of the part's and the machine's coordinate system. Next, the discretisation is performed where the goal is to obtain at each height/slice a grid with a maximum step size of 1 mm. Since the grid is equidistant in x and in y at each height, the necessary information to transform between image coordinate system and the part coordinate system can be extracted. Should the image creation be skipped, this information is still needed for following transformations, which is why it is always obtained. It must be mentioned that in the current implementation the grid starts and ends slice_dist/2 above and below the lower and upper bounds respectively. Else, the cleaning would occur at the lower boundary of the part of interest which is not desirable, leading to a higher waste of CO2. Compare Figure 4.1 for a visualisation of the difference.

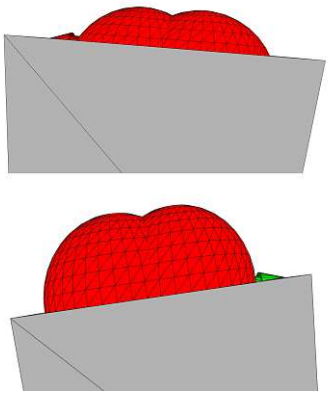

Abb. 4.1: Difference between starting the grid at the part's boundary and slice_dist/2 above. In the lower section the sphere representing a cleaning point is positioned such, that one half is outside of the cuboid. For better cleaning efficiency, meaning the lance blows less CO2 into nowhere, the start an the end height are offset from the part's boundary.

Tab. 4.1: Settings in the main program. The horizontal lines separate different types of settings.

Setting	Description
safety_offset	The distance which must be kept at all times from the part of interest.
bounds_offset	Enlarges the obtained images. Due to the safety offset it is possible that sections of the safety distance can be cut off. It is usually as large as the safety distance but should be kept as small as possible to save resources and time.
dist_points	The maximum distance between points along the contours.
slice_dist	The maximum distance between slices/heights.
part_shifts	A list containing the necessary translations in x, y and z to correct the position of the part's coordinate system to the machine's.
custom_heights	Due to the necessary shadowing it may be necessary to add custom heights. It is recommended to always add the lowest part height added with the part shift in z.
height_bounds	In case only a certain height interval is of concern the bounds can be entered here. In case they are not needed e.g. $\pm 3000\ mm$ can be used to disable this setting.
sphere_radius	The radius of the spheres shown in the visualisation. The color can change depending on the radius.
plot_bounds	Choose between part bounds or machine bounds as the bounds for the created images.
Q_folder	Q being a placeholder for a folder name. Name of the respective folder.
cleaning_inside	This setting changes how the automatic contour extraction and the determination of C angles behaves.
choose_contours	If true, the user can select which contours are of interest. Else an automatic selection, described below, is made.
create_new_images	If new images including the safety distance shall be created, this is set true. Else, already existing images are used to save time.
blend_images	If the obtained images shall be blended to obtain the shadowing.
extract_new_points	If based on the blended images new points for cleaning shall be extracted.
create_paths	If the paths between cleaning points shall be created. This leads to corner points being added to the point recipes as well as whole paths being saved for further analysis.
visualise_point_recipe	If the obtained point recipe shall be visualised.
coos	If a coordinate system shall be displayed when visualising the point recipe.
show_mesh	If the mesh of the part of interest shall be displayed.
bunny_mode	A relict which uses the stanford bunny as input. Effect similar to Monty Python's Holy Grail bunny.

To understand the necessary transformations between the part's coordinate system and the images created, it has to be mentioned that images have their origin in the upper left corner. This usually requires a translation in terms of x and y. Such a transformation can be explained by looking at two points, point (0/0) and point (x_i/y_i), both in the image coordinate system, at any given height. Point (0/0) is simply transformed by inserting its x and y coordinates, x_0 and y_0, in the part's coordinate system. For point (x_i/y_i) the respective grid distance (Δ_q) is multiplied by its image coordinates and the coordinates of point (x_0/y_0) are added, giving the position (x_p/y_p) with respect to the part reference system. This process is visualized in Figure 4.2.

Next, after the discretisation, the images, which include the safety distance from the part of interest, are created. Each grid point is used to determine the distance of the lance to the part using Open3D. The result is a tensor containing all distances which is then converted into a binary image at each height, based on the safety distance given by the user. The resulting images are saved in the 'images' folder. Figure 4.3 shows an example of one such - binary - image. Note that it would be possible to adapt the created images using dedicated software, e.g. Microsoft Paint, in case the obtained results are not sufficient.

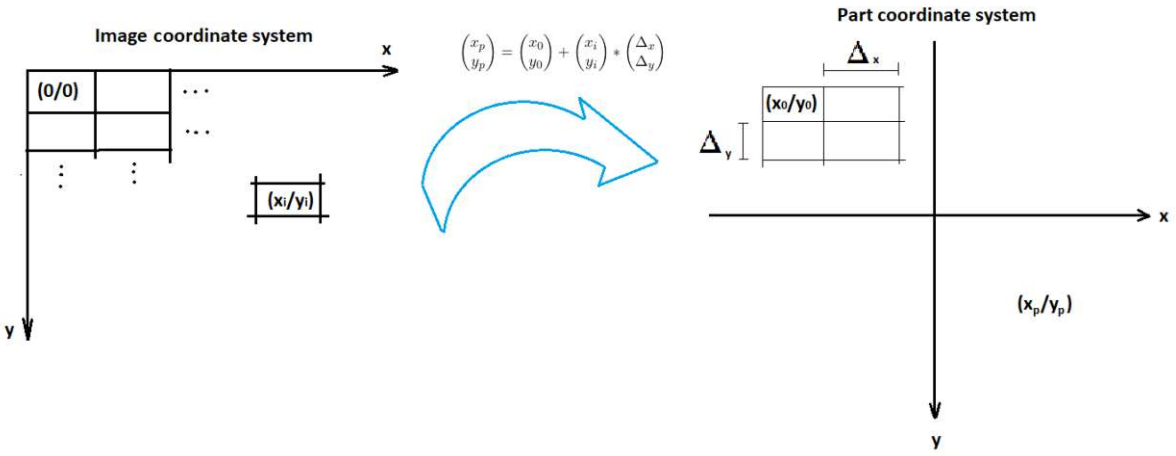

Abb. 4.2: Visual representation of the coordinate transformation between the image's and the part's coordinate system. $*$ indicates the i'th entry is multiplied with the i'th entry only.

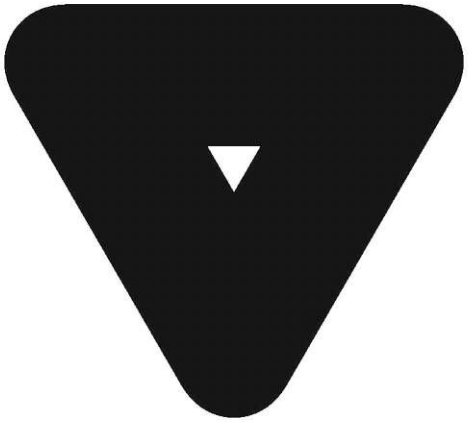

Abb. 4.3: Example for a (binary) image of the hollowed pyramid. Black pixels are grid points which are inside the safety distance and are thus not valid for cleaning points and for pathing.

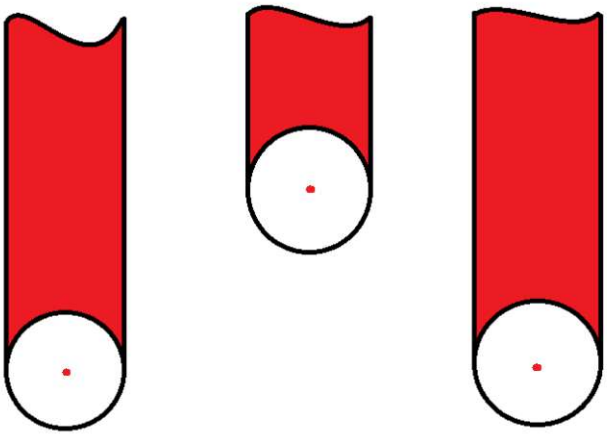

Abb. 4.4: Depiction of the shadowing (red, above). The lance of the machine cannot bend around obstacles, as shown here with the three projected lines. Thus, not only the distance at one height has to be considered but also obstacles which lie below a particular height. This is achieved by blending images from below onto images from higher slices.

Due to the design of the cleaning apparatus with it's long lance, the shadowing from structures lying below must be considered to avoid collisions at heights lower than the current cleaning layer. Figure 4.4 shows an image of three projected lines in 3D space. Around all lines a safety distance (white circle) in all directions is shown, which must be kept at all times. Yet, the lance cannot bend around the safety distance and thus the restricted area (red, above) must be projected onto higher slices. The fact that inclined surfaces or features in the part of interest can influence higher slices makes it necessary to introduce custom heights. These heights can be defined by the user to create images, and thus shadowing, at these heights without the extraction of cleaning points.

To obtain the shadowing, all images of lower heights are blended onto images of higher heights. Blending refers to adding two pictures with defined weights and a possible offset, leading to an overlay of both images. In this application, both images are weighted by 0.5 with no additional offset. Finally, a binary image is created from the overlay and saved into the 'blended' folder. All images which originate from custom heights are deleted to not appear in following steps. The application of this method cannot recognize eventual orifices meaning that in the end the algorithm could place points where no structure exists.

Following, the contours contained in the blended images are extracted. To achieve this, the function findContours from OpenCV is used with RETR_TREE and all contour points stored. RETR_TREE sorts the extracted contours by hierarchy. To facilitate the extraction of contours two different modes were implemented into the main program:

1. Shows the user a depiction of all extracted contours and highlights in green one specific contour, which the user can choose to include or not, see Figure 4.5. Upon selection (by pressing y or n) the algorithm continues with one important aspect: if the same number of contours are detected, it is assumed that the sliced part has the same structure as in the previous height and thus the same contours (as sorted by RETR_TREE) are chosen. This lowers the amount of choices which have to be made by the user, especially for large parts, provided the part of interest meets the assumption taken.

2. Automatically chooses the contours. Since all contours are sorted by hierarchy, it can be differentiated between outside and inside contours. When setting the cleaning_inside flag to True, the outer contours, which are always the first ones, at position 0, are ignored and all other contours are selected. It has to mentioned that in this mode it is assumed that the slices of the part of interest can be differentiated into inside and outside. This breaks down if, e.g., 'legs' or 'arms' are part of the structure, leading to the first leg or arm being ignored and all others being selected. When the inside_cleaning flag is set to False the contour at position 0 is considered and all other are ignored. Similar restrictions are of course valid as to when the cleaning_inside flag is set to True.

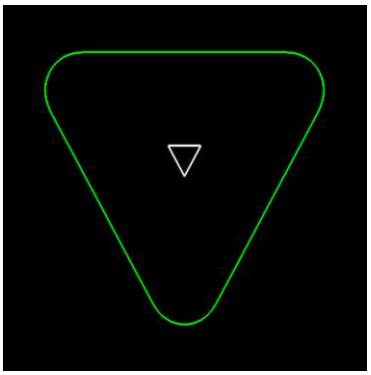

Abb. 4.5: Example of the contour selection window. The current contour is shown in green, all other contours are shown in white. By typing y or n the user decides which contours are of interest for the point extraction. Slices where the same number of contours are detected as in the previous selection lead to an equal selection of contours.

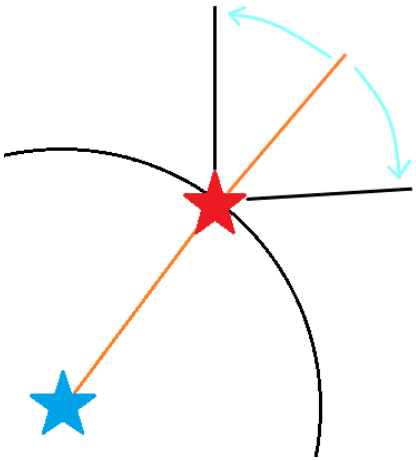

Abb. 4.6: Visual representation of how C_1 and C_2 are determined in the current implementation. A reference point (blue star) is determined, assuming enough points are available, via a fitted ellipse. At each point a line is drawn (orange) towards the reference point (red star) and at each side one 'half-range' (teal) is added. In case angles larger or smaller than $\pm 180°$ are obtained, an additional point is added which covers the residual.

Next, follows the creation of the csv file for each height. Due to their similar concept, dataframes are used inside the program. All points of the considered contours are placed into the dataframe with their x, y and z coordinates, as well as their coordinates in the image coordinate system and the contour id they belong to. The contour id is an integer which helps in assigning points to contours in the next steps. Following, the points inside the dataframe are selected and converted to points at which the apparatus stops to clean. The selection follows the following idea: if the total length of a contour is long enough, points are set at equal distance along the contour. To achieve this, the total length of the contour is divided by dist_points, before applying the ceiling function to the result. This ensures that dist_points is never exceeded and a denser point cloud is obtained. In case the contour length is smaller than dist_points, only the first point in the contour is set as a cleaning point.

To obtain a full point recipe, the C and R angles have to be obtained. To achieve this, an algorithm called polar_centre_macro from an existing implementation was adapted for higher performance and better readability in the code. To determine the C angles it uses a reference point from which a line is drawn to the point in question. To the angle connecting the two points, the so-called half-range is added and subtracted. The resulting angles serve as the C1 and C2 entries in the point recipe. A visualization of the process is shown in Figure 4.6. The value of the R angles and the half-range are constants and in the presented implementation they are based on a recommendation of the previous operator of the cleaning apparatus.

To obtain the reference point for each contour, fitEllipse from OpenCV is used. It has to be noted that if the contour is not convex, there is the possibility of the algorithm failing since the reference point could lie outside of the contour, see Figure 4.7. It has to be noted that the fitting of the ellipse requires at least five points. In case fewer points are available, it is assumed that the user will have to intervene anyways and thus the coordinates of the first point of the contour are chosen for the reference point.

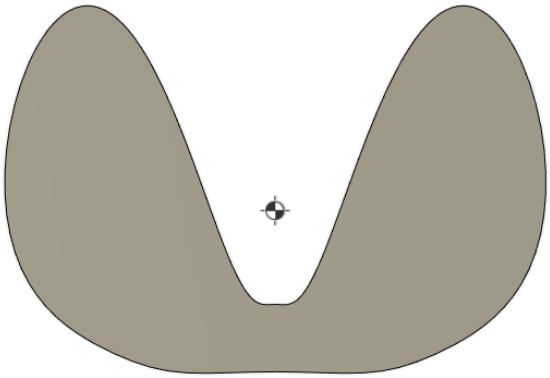

Abb. 4.7: One possible example, where the reference point could lie outside of the contour, leading to a failure of the algorithm. The geometric center represents a possible reference point.

After adding the position of the reference points to the point recipe, the C and R angles are determined by the above described scheme. Finally, all points are sorted by θ, the angle of the line connecting the reference point and the point in question, the distance to the reference point and the absolute value of C1. This terminates the point extraction with a complete point recipe for all heights containing all x, y and z coordinates as well as C and R angles.

4.3 Pathfinding

After obtaining the cleaning points, where a cleaning procedure is performed, the pathfinding between these cleaning points is done. It is described as psudocode in Algorithm 2. As mentioned above, the pathfinding library uses graphs in the form of arrays/matrices to identify obstacles or accessible positions. Providing such a matrix, in this case a blended image, with a start and an end point, which have to be on acessible points, is already enough to start the procedure. Note that the for-loop in line 4 is reversed to facilitate the insertion of the corner points.

Whole paths are separate csv files containing all positions between start and end points found by the A* algorithm. They are used to confirm at each position along the path, at a maximum distance of 1 mm, what the distance between the lance and the part of interest is.

In some cases it was observed that the A* algorithm induced many corners in the path, see Figure 4.8. This is not desirable since the cleaning apparatus has to turn in the direction it is moving which can take some time. Instead, the algorithm should come up with as few corners as possible to reduce the travel time between points. As a result, 'highways' were conceived.

The idea is based on the fact, that the A* algorithm uses weights to determine the cost to get from one point to the next. As such, the weights in rows and columns where a start and end point are located are reduced[1], leading to the algorithm sticking more

[1] Only where no obstacle is located.

Algorithm 2 Create Path

for $height = 0, 1, \ldots, n_h - 1$ **do**
    Load blended binary image.
    Multiply binary image by f.
    for $point = n_p - 1, n_p - 2, \ldots, 1$ **do**
        Get start and end point.
        Decrease weights along rows and columns of start and end points.
        Run A* algorithm.
        Reset lowered weights.
        if $len\_path \geq 0$ **then**
            Find corner points in path.
            Transform corner points from image coordinate system.
            Insert corner points into dataframe.
            Transform and append whole path points.
        end if
    end for
    Write csv files.
end for

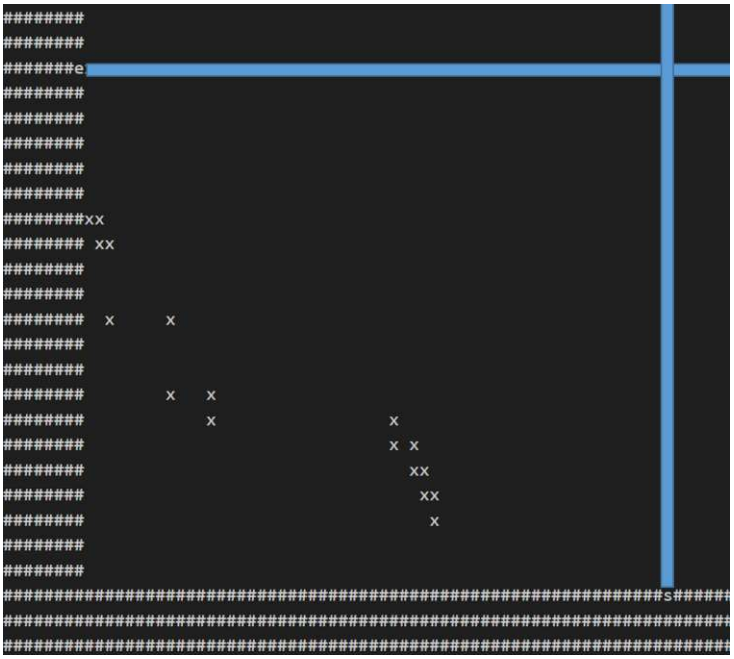

Abb. 4.8: Visualisation of the pathing using the pathing library. # are obstacles, s and e depict the start and endpoints. x depict corner points. The blue lines show where the 'highways' with lower weights are running along. As can be seen, the unmodified version creates many corner points which prolong cleaning operations since the lance has to turn in the direction where it is moving.

to these lanes where movement is cheaper, see also Figure 4.8. Note, that the lanes are removed before the next path is searched for.
The multiplication factor f, used to increase the weights globally, in Algorithm 2 was determined to be sufficiently high at 2. Even much larger values did not lead to further decrease in the number of corner points.

4.4 Visualization and Virtual Validation

Several tools with different use cases were applied to be able to validate a point recipe visually and numerically.

- One basic tool is CR_plotter. It can visualize point recipes in 3D as points and display the position in the point recipe. A switch enables the visualisation of the C and R angles as cones. It must be mentioned that the cones are only a tool for visualisation and do not represent the actual dimensions of the CO2 jet. Also, the tool was developed for an older version of point recipes, where the index starts at 1 instead of 0.

- point_visualisation uses Open3D to visualise single csv files created in the extraction process.

- single_collision_point uses Open3D to visualise the part of interest together with the lance at a position selected by the user. It also returns the minimum distance between the lance and the part of interest and also returns the involved points of the lance and the part respectively. A red line connecting the two points is also visualised.

- collision_control acts similar to the previous tool but does not render the situation at hand. It takes a point recipe and plots barplots containing the distances at points in a csv file. A flag is used to switch between the point recipe, containing cleaning points and corner points, and whole paths which contain the position of the lance at each grid point along its way. To avoid human error, one Jupyter cell checks if all points fulfill the safety distance and prints a statement depending on if all positions keep the required safety distance.

- Inside the main program another visualisation tool is embedded ,which takes all created csv files and displays them, if chosen, with the part of interest and a coordinate system for better orientation.

4.5 Tree builder

It is possible that parts of interest can have different cavities. To be able to assign each contour with the right cavity a script called tree_builder was written. It uses the extracted contours and the reference points together with OpenCV's pointPolygonTest to determine which contour is connected to which. It is a temporary implementation which has to be uncommented in the main program. It is only mentioned in this document for completeness.

4.6 Suggested Usage Process

Due to the various tools and the complexity of the algorithm a suggested process shall be outlined here. The assumed situation can be summarised as follows: a new part to be cleaned, with the only information given beforehand being the necessary translations which have to be applied, and all folders which are filled with data during the process are empty (Images, blended, csv and whole_paths).

1. Let the algorithm run with no new image creation, no blending, no new point extraction, visualization being optional to get the bounds of the part of interest and to get a grasp what the part looks like.

2. Add the lower bound of the part as a custom height and let the algorithm run with enabled image creation, blending of images and new point extraction. Optionally, also enable the visualization of the point recipe to see if the results are as expected. At this stage the csv files can be manipulated to move, add or delete points from the point recipe.

3. Validate the safety distance of all extracted points by using the collision control Jupyter notebook. In Case the safety distance is violated, use the single collision point script to get a grasp of the situation. Possible solutions include adding or changing custom heights, moving the point in the csv file, or increasing the safety distance in general.

4. If necessary, use tree_builder in the pathing module to see which contour connects to which. Based on that insert:

   - At the first height to be cleaned (typically starting at the top) insert before the first point to be cleaned a point with the same coordinates as the first one but at $z = 0\,mm$. This ensures that the machine does not collide with the frame when going to the first point.
   - At each change in height, set before the first cleaning point a point with the same coordinates as the last point of the previous height but at the current height[2]. This ensures at the whole path creation that these height changes are also incorporated.

5. Create the paths/whole paths and check at all heights that the safety distance is kept.

6. Stack the point recipe as needed. In case of an inside cleaning follow the recommendation given by the graph_builder. In case of an outside cleaning, stack all contours with index 0.

4.7 Runtime Measurements

As mentioned above, the determination of the C and R angles is based on an existing implementation. The new implementation makes heavy use of the pandas library, making the code more readable and faster. To determine the performance a set of randomly

[2]Consider that at the previous height at the last cleaning point a violation of the safety distance cannot occur. Thus, this position is used to change height.

generated points was used. The randomness was generated using Numpy which uses a 64 bit Permuted Congruential Generator to obtain the numerical values [28, 29]. The module itself was tested outside of the main program for its performance. For each set of points $(10^0, 10^1, 10^2...)$ 20 runs were performed. If the execution took longer than one minute only five runs were performed and no larger set of points was used.
To determine the behaviour of the main program with increasing part size, different versions of the cuboid (see next chapter) were created and loaded into the algorithm. The idle run had all switches set for no creation of images, blended images, csv files and whole paths and loaded the cuboid_300. Each case was repeated five times, visualisation was always turned off and all folders which contained files created by the algorithm were deleted before each execution. Recorded were the times needed until the point extraction finished and until the pathing (referred to as complete) was finished, making the point extraction time part of the complete times.
The laptop used for the measurements runs on Windows 10 Pro with an Intel Core i7-8665U with 16 GB RAM and is x64-based.

5. Validation

To be able to validate the algorithm, some simpler geometries were created. One more complex structure was chosen to serve as a more realistic use case, which was also mounted onto the cleaning apparatus for practical runs (REQ5). After the application of the main algorithm, the tools for visualisation and validation were applied and are presented in this section.

5.1 Applied Geometries

The algorithm was applied to several simpler geometries, which served as a testing ground:

- Cuboids of different size (see Figure 5.1).
- A hollowed pyramid structure (see Figure 5.2).
- A funnel structure (see Figure 5.3).
- One hexagonal plexiglass structure.[1]

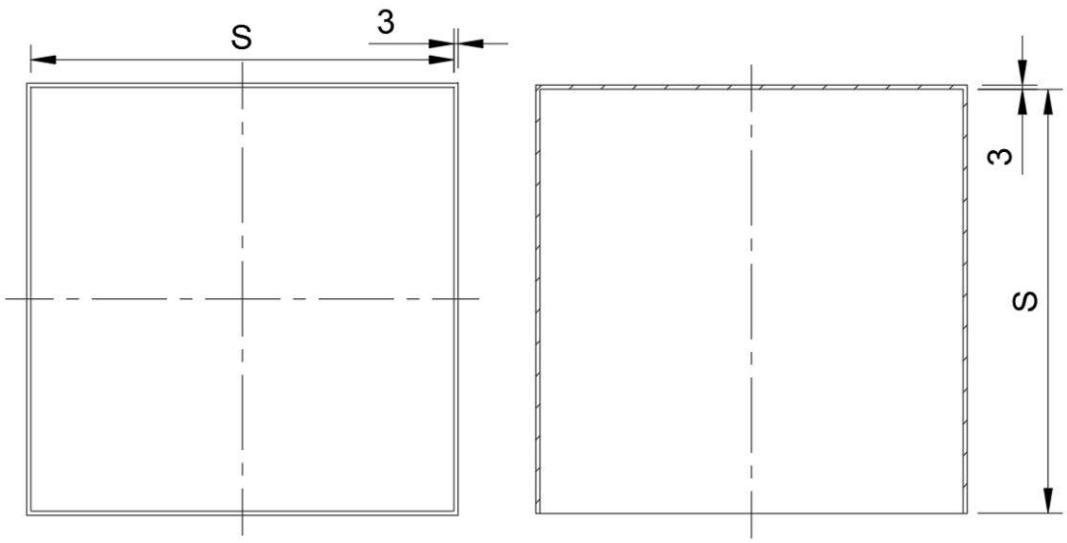

Abb. 5.1: Sketch of the used cuboid structure. S is a placeholder for the different versions: 300, 600 and 1200 mm.

[1] Since this component resembles an actual component, only the results are shown, not the actual structure.

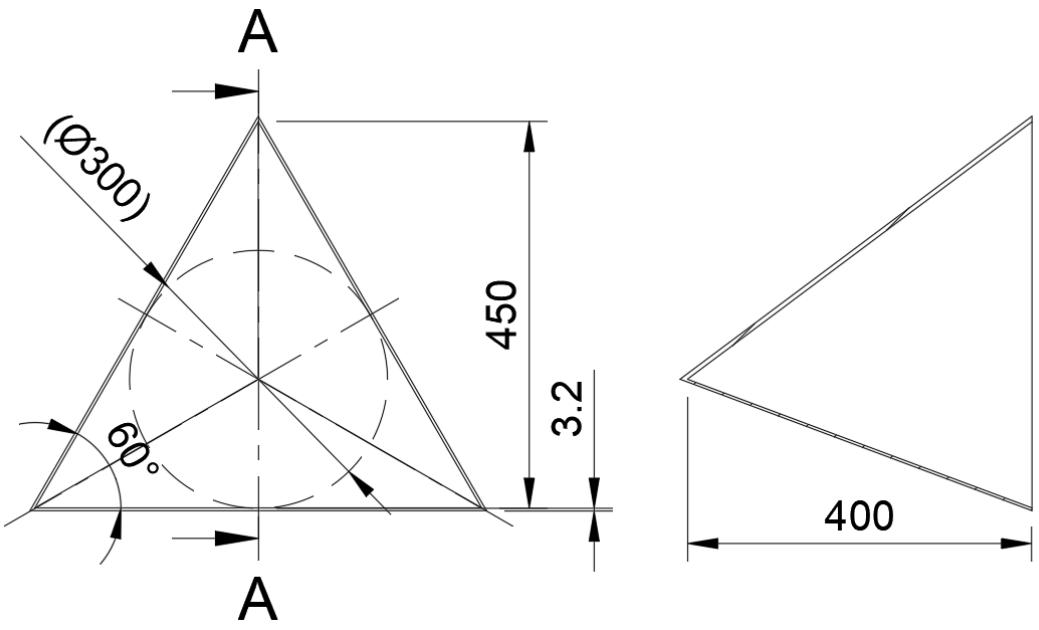

Abb. 5.2: Sketch of the hollow pyramid.

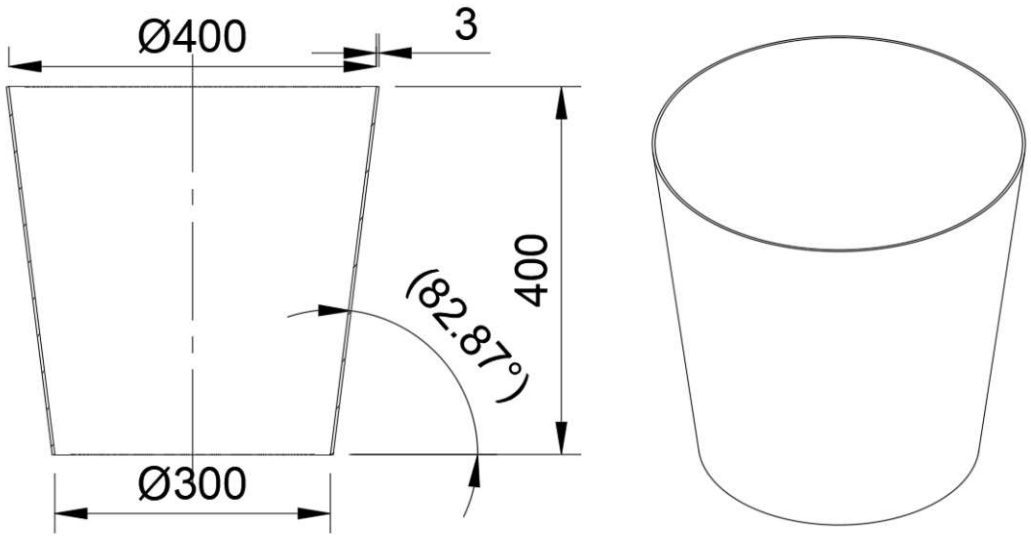

Abb. 5.3: Sketch of the funnel.

The hexagonal structure was mounted onto the machine to serve as a practical validation of the algorithm (REQ5). If not mentioned differently Table 5.1 shows the applied settings for the main algorithm. For the hexagonal structure the settings shown in Table 5.2 were applied. In comparison to the other geometries the Safety_offset was increased to gain additional security in the practical run. slice_dist was changed to get a more even distribution of points since it was intended to use the obtained point recipe for an evaluation of the cleaning efficiency. The parts_shifts values were determined using the laser distance sensor of the machine, (see Figure 5.4). custom_heights stems from the lower z boundary of the stl file, parts_shift[2] refers to the applied translation to the structure in z, essentially translating the lower boundary together with the rest of the structure. The height bounds were chosen to obtain a total of 13 slices,

which was only part of the whole structure. Due to the symmetry of the structure only one panel was chosen for the validation runs.

Tab. 5.1: Applied settings for the main algorithm.

Setting	Value
safety_offset	100 mm
bounds_offset	100 mm
dist_points	80 mm
slice_dist	100 mm
custom_heights	[]
sphere_radius	100 mm
plot_bounds	part_bounds
cleaning_inside	True
choose_contours	False

Tab. 5.2: Applied settings for the main algorithm for the hexagonal plexiglass structure. If not mentioned the settings in Table 5.1 are applied.

Setting	Value
safety_offset	110 mm
bounds_offset	110 mm
slice_dist	80 mm
part_shifts	[0.0, 31.765, 36.0] mm
custom_heights	[72 mm + part_shift[2]]
height_bounds	[-3000, 1107] mm

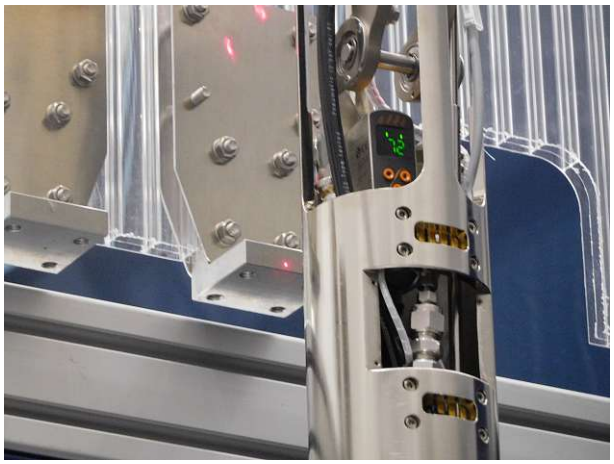

Abb. 5.4: Measuring the difference between the machine's and the part's coordinate system. The laser distance sensor in the lance also displays the measured distance, allowing for a correction. To achieve this, points in the point recipe are selected, carefully navigated to and the value on the display is read. Note, that the distance measured is lower than the minimum safety distance. This is explained in Figure 5.5. The other laser points in the image show the orientation of the CO2 jet nozzle.

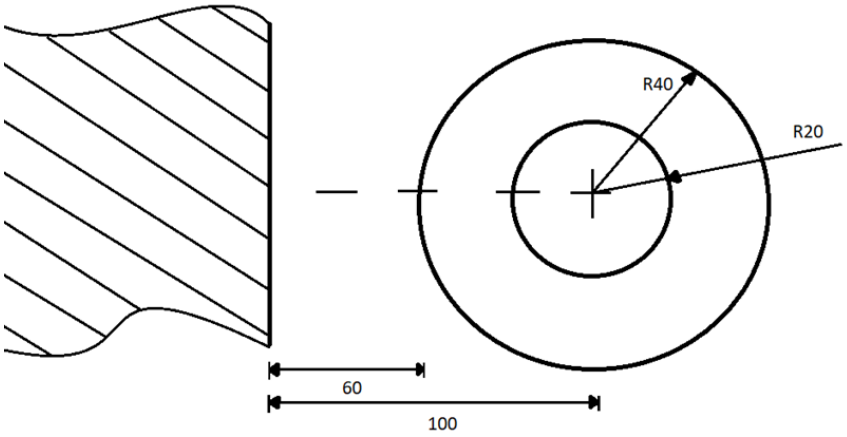

Abb. 5.5: Sketch of how the safety distance differs between the laser distance measuring device and the input in the algorithm. In the algorithm it is always referred to the center of the lance at the height of the nozzle. The CO2 nozzle is offset from the center of the lance by 20 mm (inner circle), the laser distance sensor by 40 mm (outer circle).

The tools for visual and numeric validation were applied to the above shown geometries. Figures 5.6 to 5.20 show the resulting visualisations and plots. Figures 5.6 to 5.11 show the results for the different cuboids. Figures 5.12 to 5.15 show the results for the hollow pyramid and 5.16 to 5.20 show the results for the funnel. If no variation was observed between slices, the results are not shown in this document.
For the hexagonal structure, Figures 5.21 to 5.31 show the visualisations and plots of the obtained points. For the hexagonal structure no variation was observed for slices higher than the third one, which is why only these slices are presented.

The point recipes of which the plots are shown in the following chapter received no modification as suggested in step 4. Only the point recipes for the plexiglass structure were modified according to step 2 since at height 0 and 1 the algorithm did not extract the points as intended due to 'legs' of the structure. Also, due to the symmetry of the structure, only one panel was used for the validation and all points lying at a position of $y > -660\,mm$ were removed.

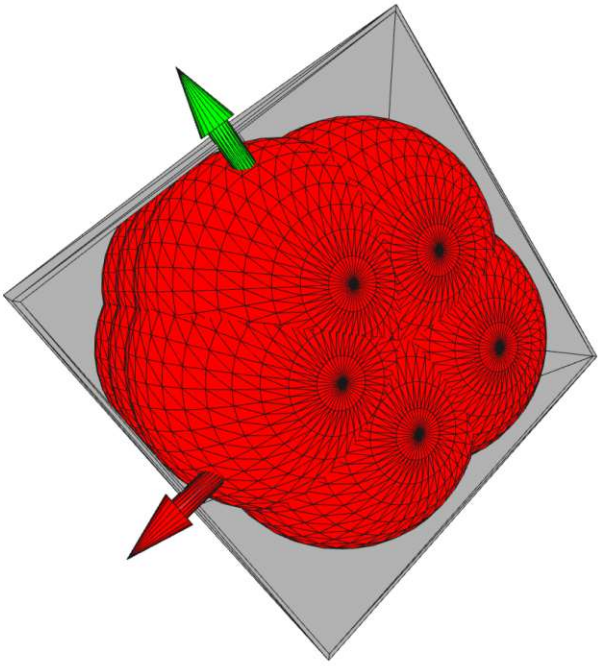

Abb. 5.6: Visualisation of all extracted cleaning points for the cuboid_300. The red spheres have a radius of 100 mm and have their center at the coordinates of the cleaning points, which represent the center of the lance at the height of the CO2 jet nozzle.

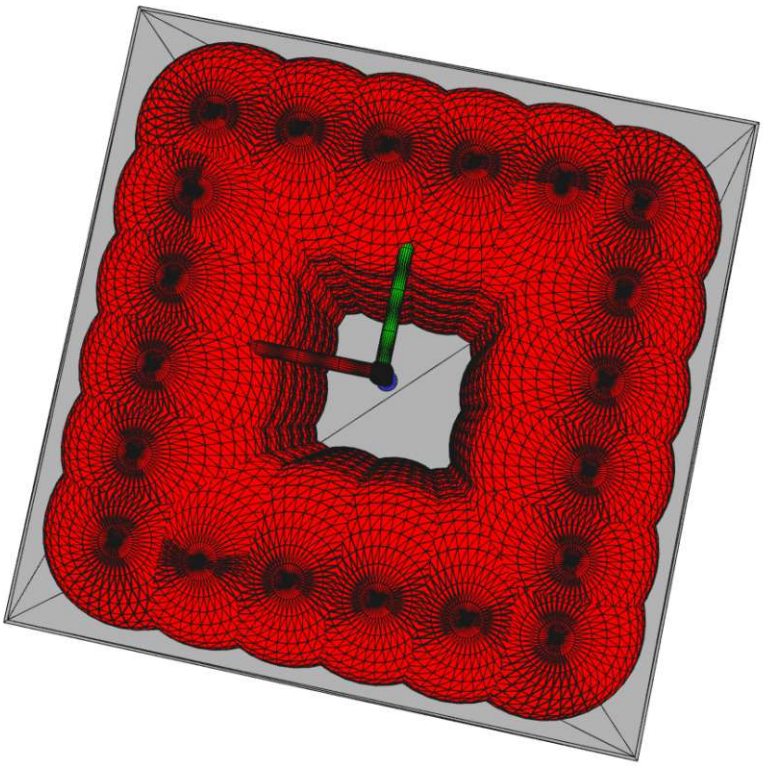

Abb. 5.7: Visualisation of all extracted points, including the corner points, for the cuboid_600. Corner points are responsible for the darker edges, since they are around 1 mm from the cleaning points.

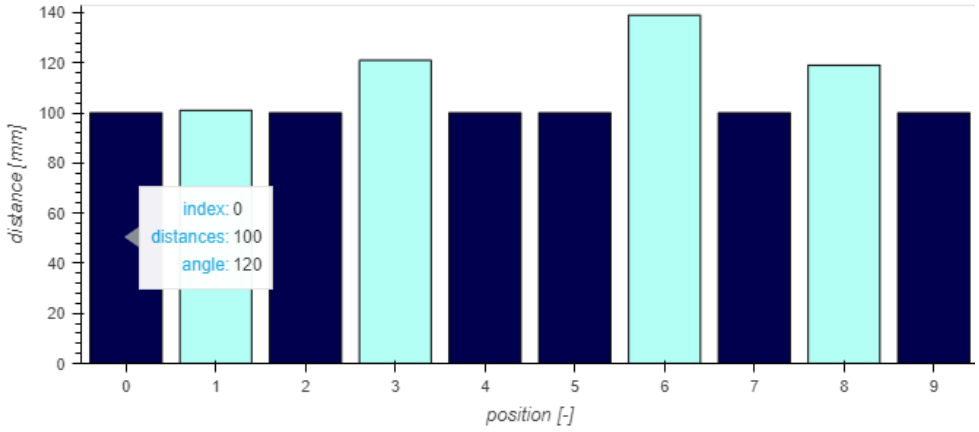

Abb. 5.8: Visualisation of the distances for all cleaning points and corner points for the lowest slice of the cuboid_300. Note the interactive display when hovering over the graph. The shown angle, in reality R1, is used to differentiate between cleaning points (dark blue) and corner points (light blue), where no cleaning program is executed, since in the current implementation corner points have R1 set to 0°.

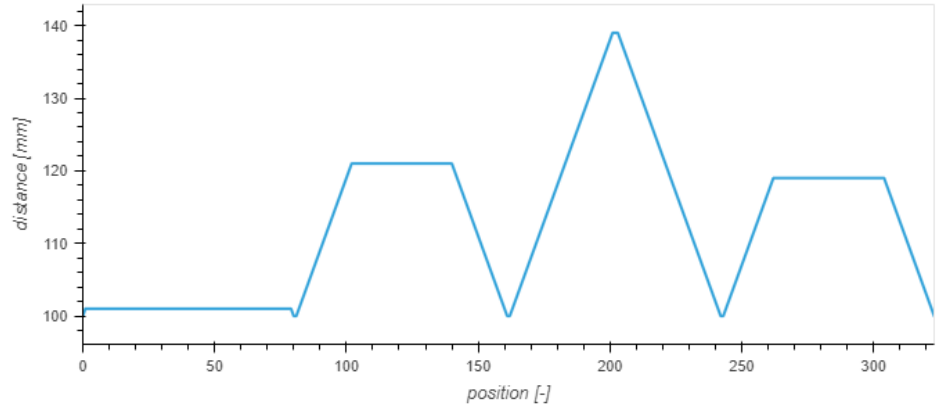

Abb. 5.9: Visualisation of the whole path of the lowest slice of the cuboid_300. Note that whole paths are saved in a different csv file, resulting in different positions in said csv file for the same points as displayed in Figure 5.8

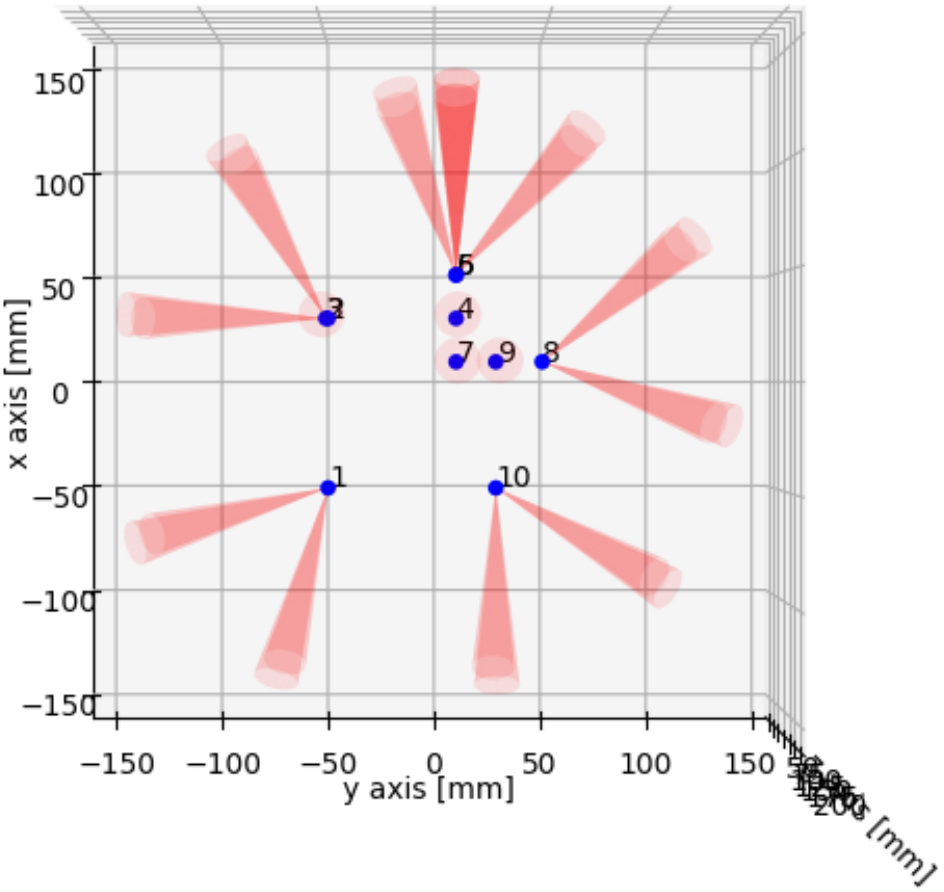

Abb. 5.10: Visualisation using the CR_plotter for the lowest slice of the cuboid_300 looking in -z direction onto the xy-plane. The position for each cleaning point and corner point is given, as well as a visualisation of the area covered by the CO2 jet (red cones). Note the slightly darker cone at position 5 and 6. Here, the C angles go further than $\pm 180°$ which is why two points are placed at the exact same position (see also Figure 5.8). The red circles at positions 2, 4, 7 and 9 can be derived from $R_1 = 0°$.

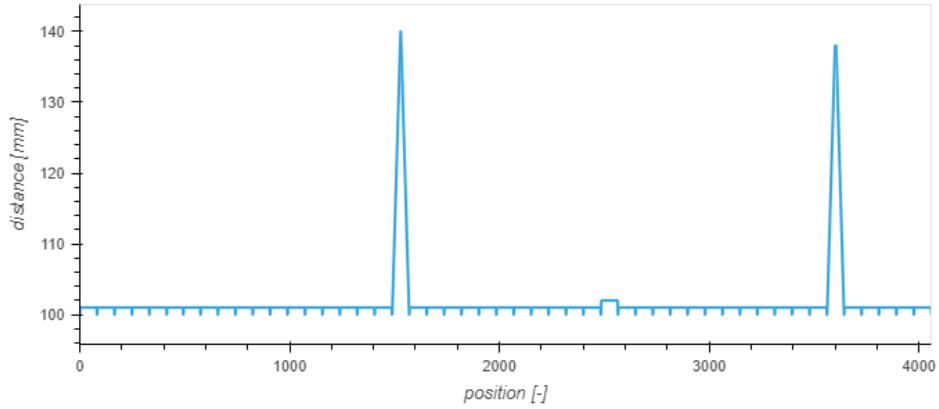

Abb. 5.11: Line graph of the whole path for the cuboid 1200. Note, how the cleaning points can be distinguished from other positions.

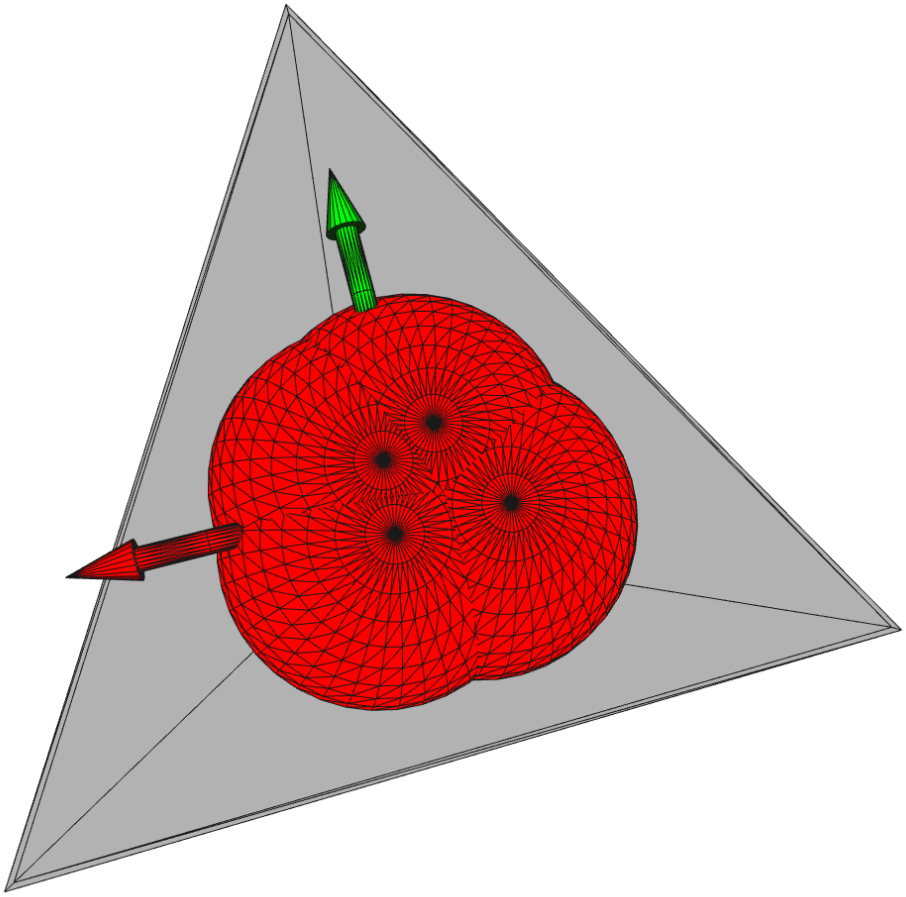

Abb. 5.12: Visualisation of the extracted cleaning points for the hollow pyramid. Only one csv file containing points was created.

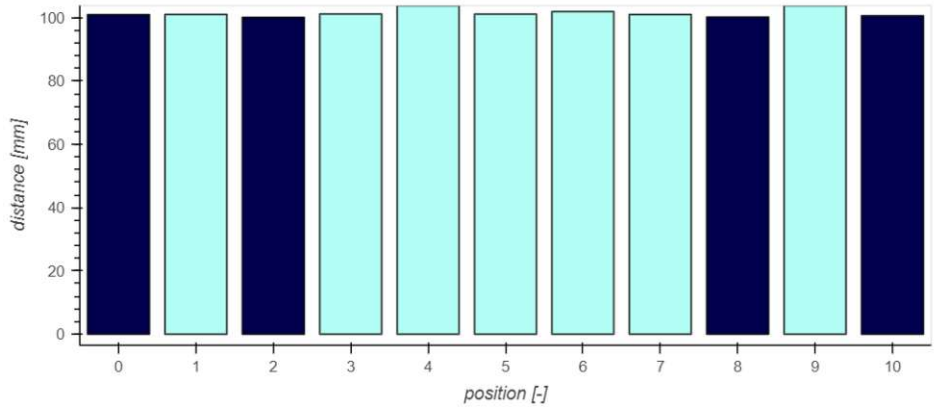

Abb. 5.13: Barplot of the distances for the cleaning points and the corner points for the hollow pyramid.

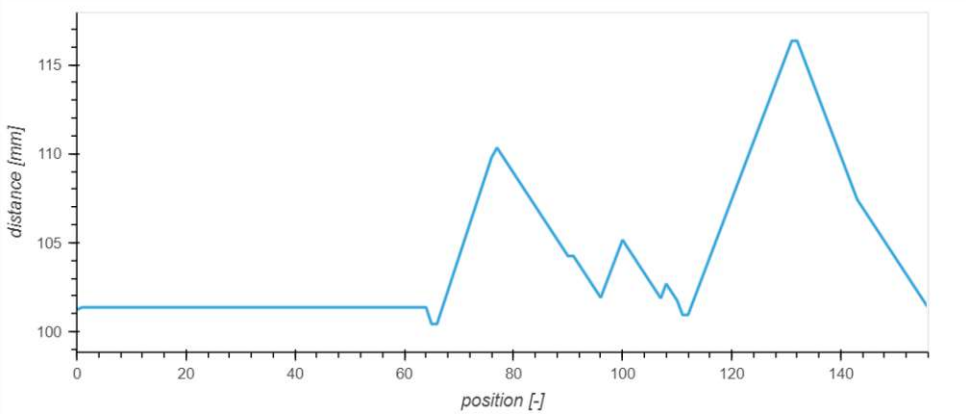

Abb. 5.14: Line path for the whole path for the hollow pyramid.

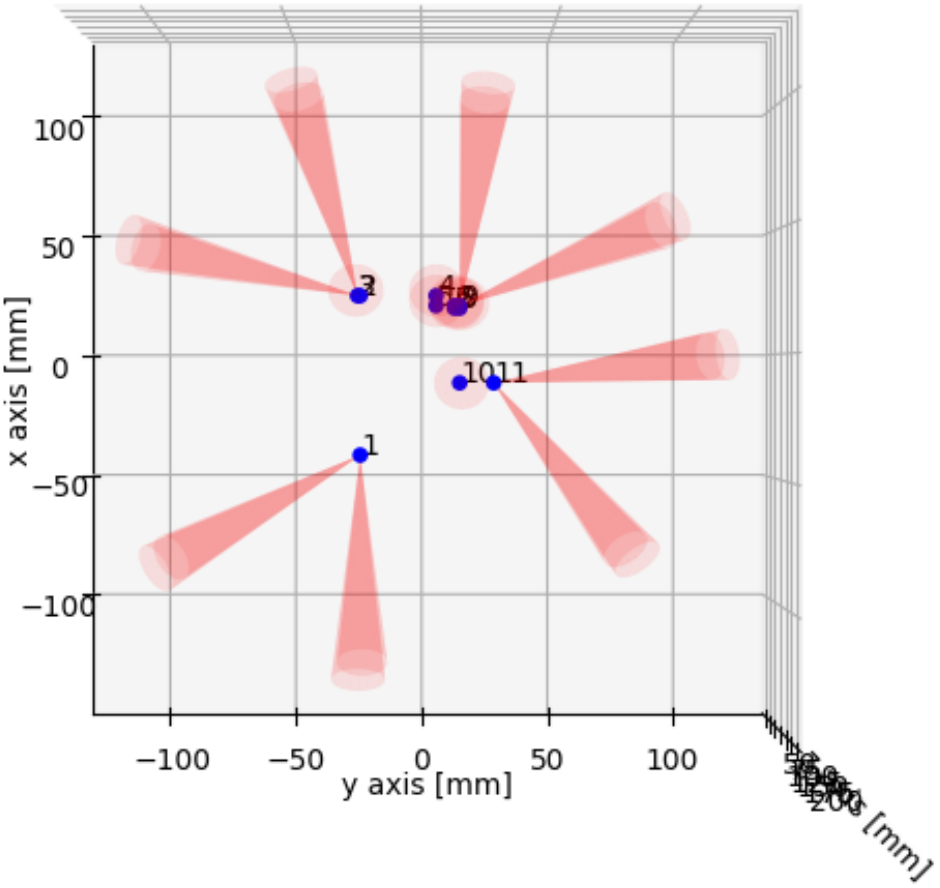

Abb. 5.15: Visualisation using the CR_plotter for the hollow pyramid looking in -z direction onto the xy-plane.

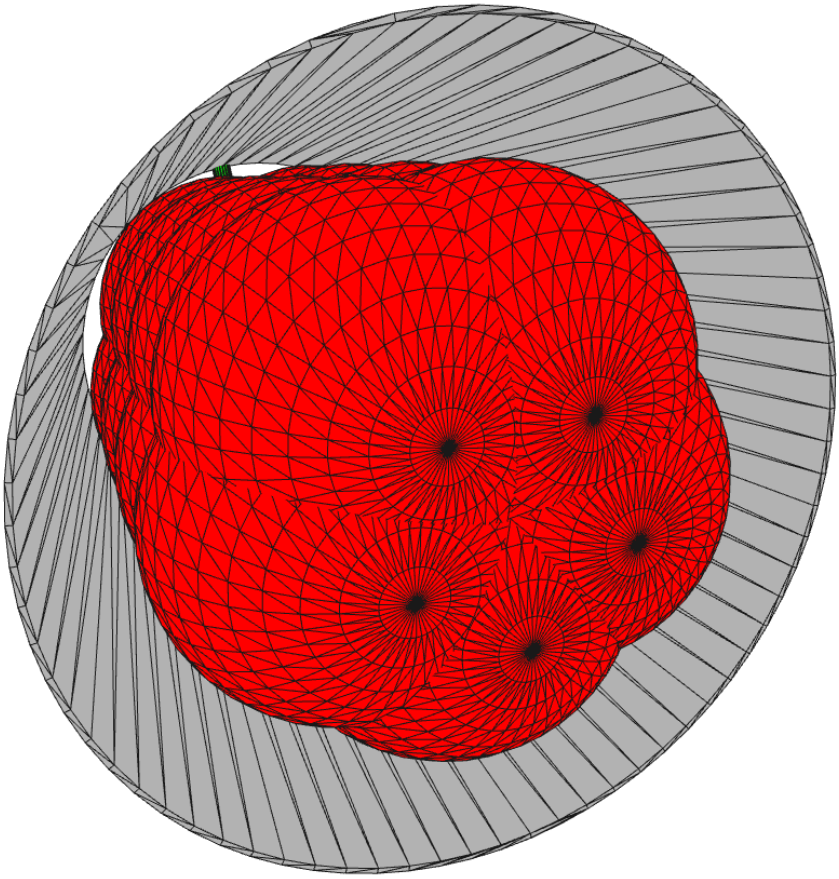

Abb. 5.16: Visualisation of the extracted cleaning points for the funnel. Note the close position of the spheres (100 mm radius) to the funnel.

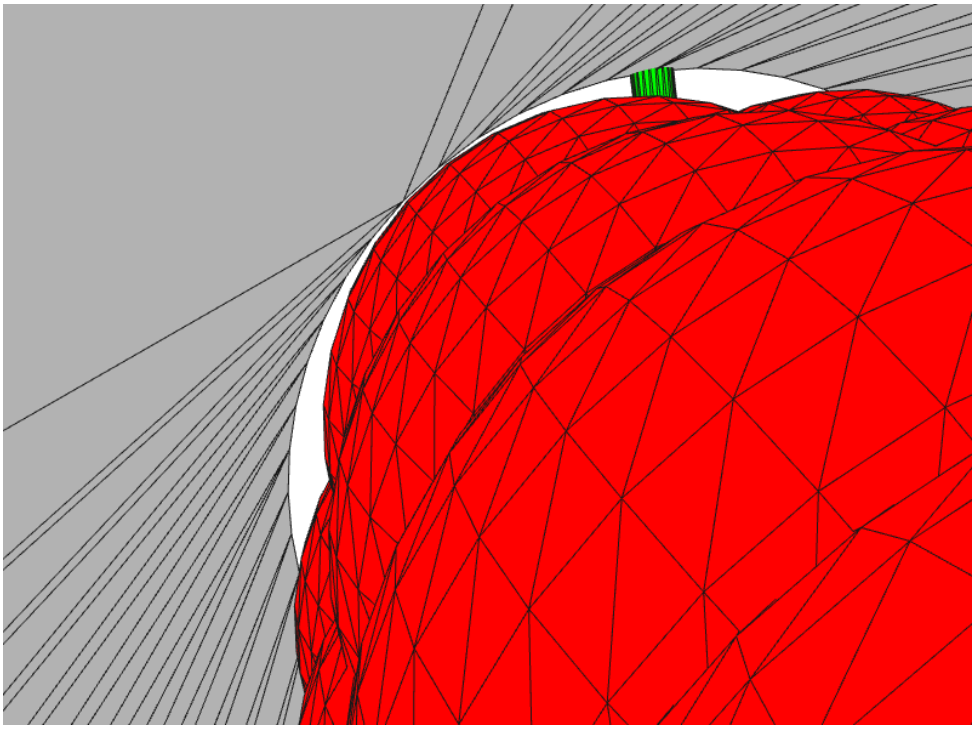

Abb. 5.17: Detailed visualisation of the cleaning points for the funnel.

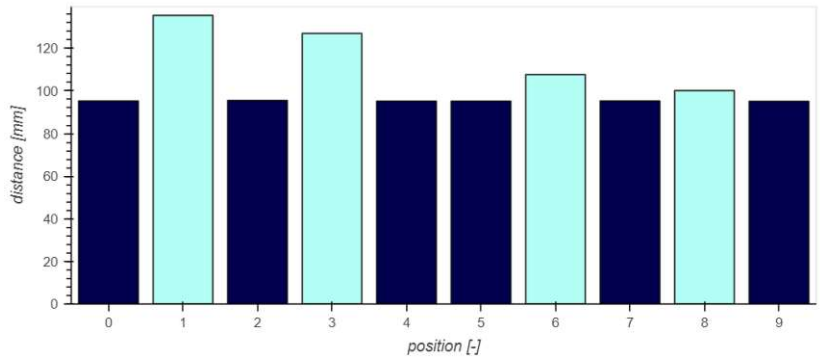

Abb. 5.18: Barplot for the extracted clenaing points and corner points for the funnel. Note that the safety distance was not kept.

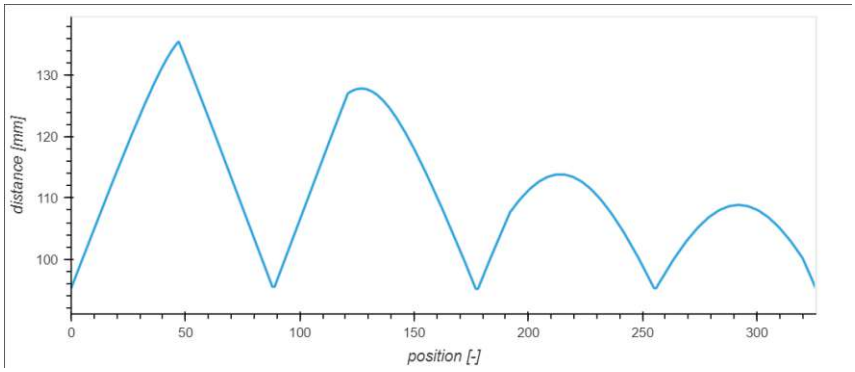

Abb. 5.19: Line plot for the whole path of the funnel.

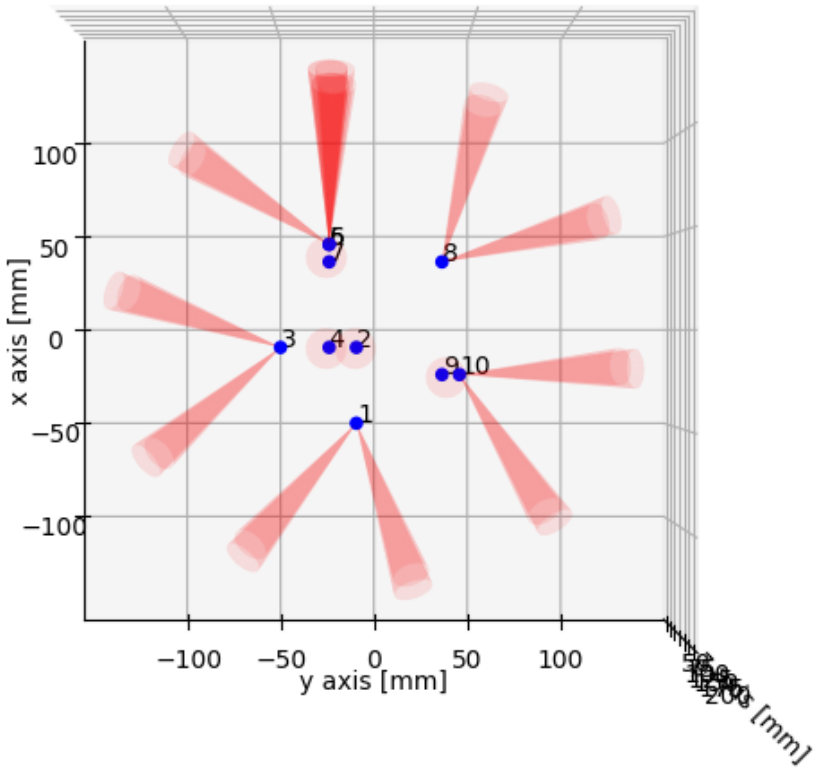

Abb. 5.20: Visualisation using the CR_plotter for slice 0 of the funnel looking in -z direction onto the xy-plane.

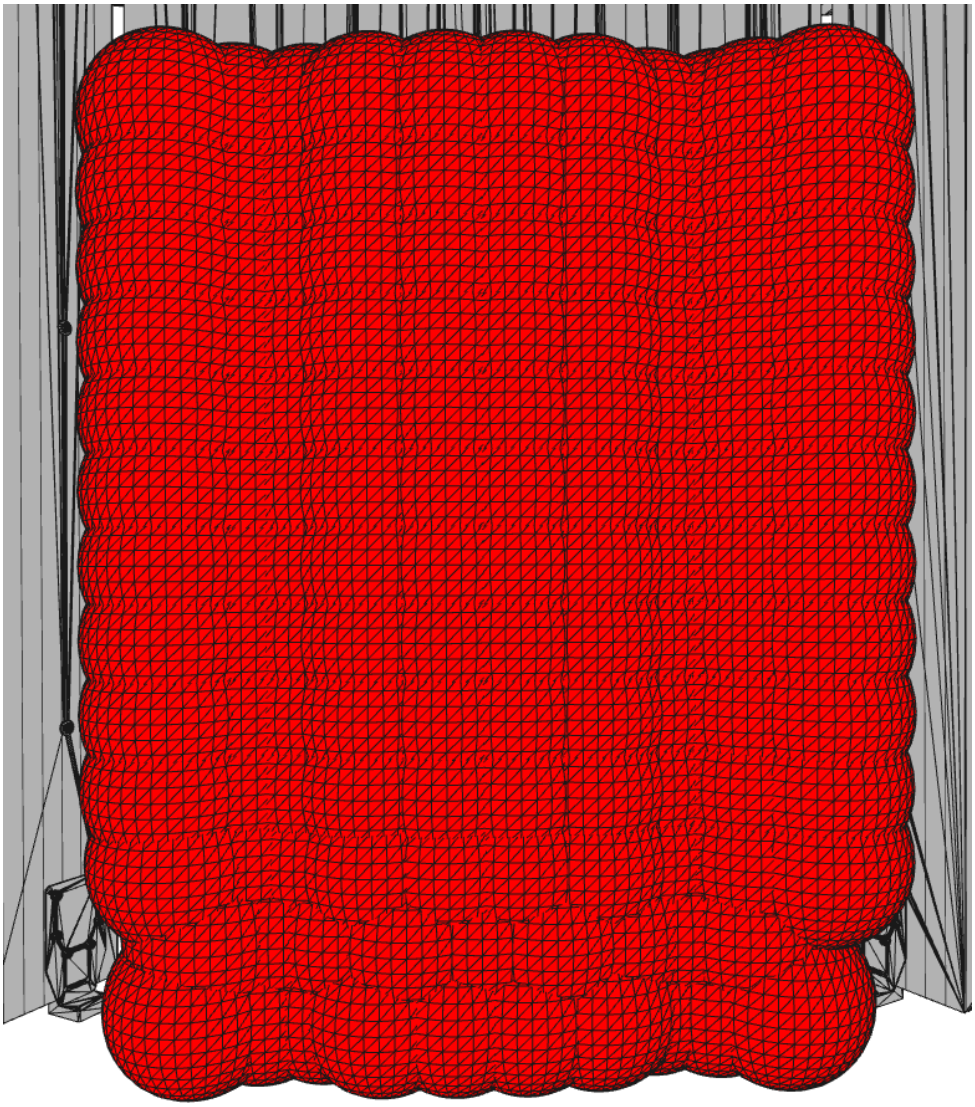

Abb. 5.21: Visualisation of the extracted cleaning points for the plexiglass structure.

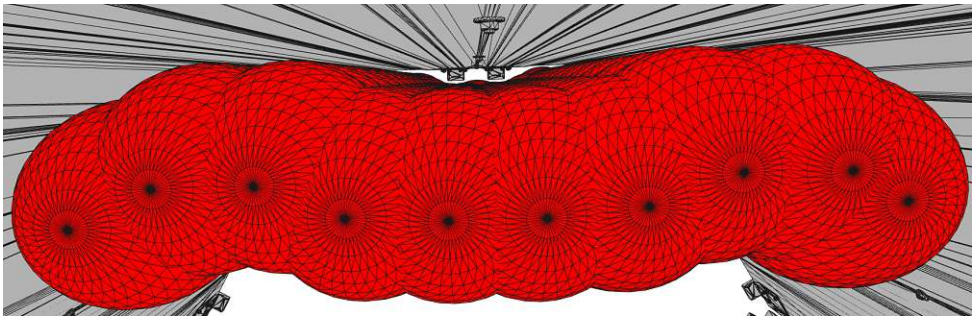

Abb. 5.22: Visualisation from above of the extracted cleaning points for the plexiglass structure.

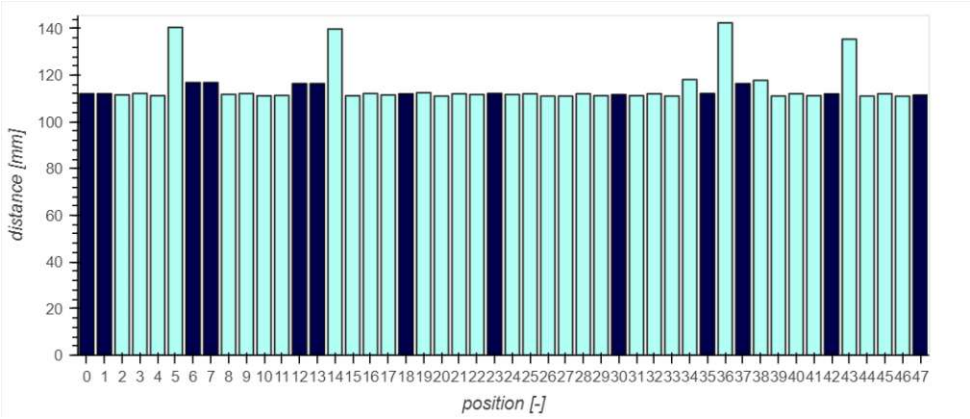

Abb. 5.23: Bar plot for cleaning points and corner points of slice 0 of the plexiglass structure. Note, how some of the cleaning points are positioned significantly further than the 110 mm set for safety_distance, and some corner points sitting slightly lower than the set safety_distance.

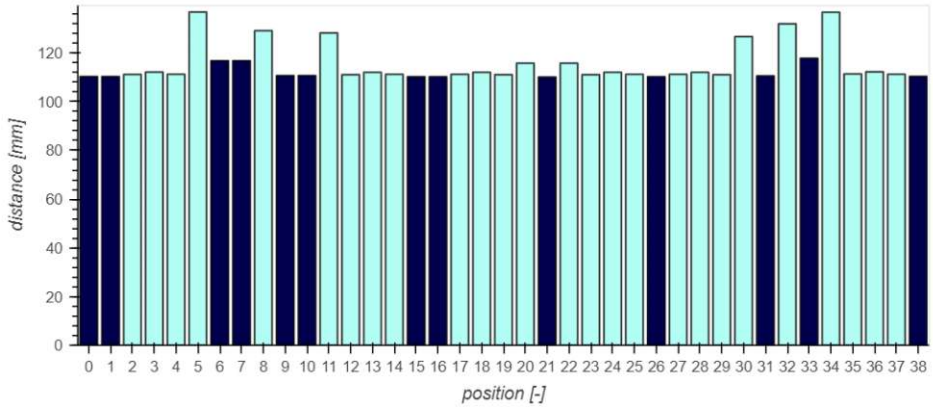

Abb. 5.24: Bar plot for cleaning points and corner points of slice 1 of the plexiglass structure. Note, how some of the cleaning points are positioned significantly further than the 110 mm set for safety_distance, and slightly lower than 5.23.

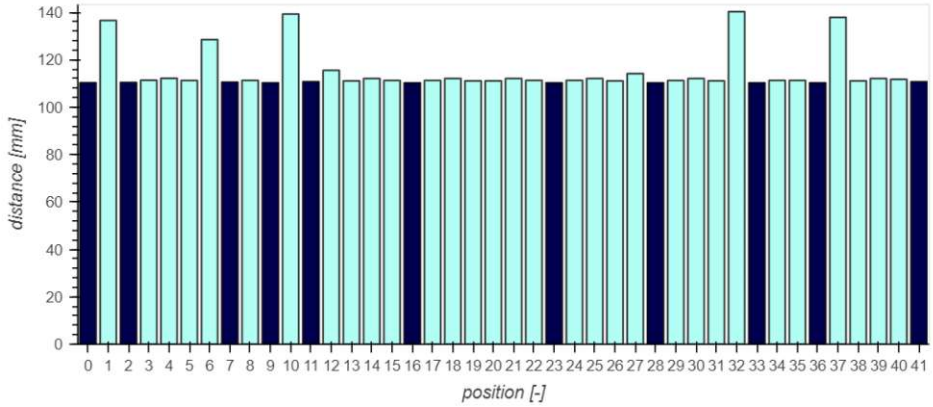

Abb. 5.25: Bar plot for cleaning points and corner points of slice 2 of the plexiglass structure.

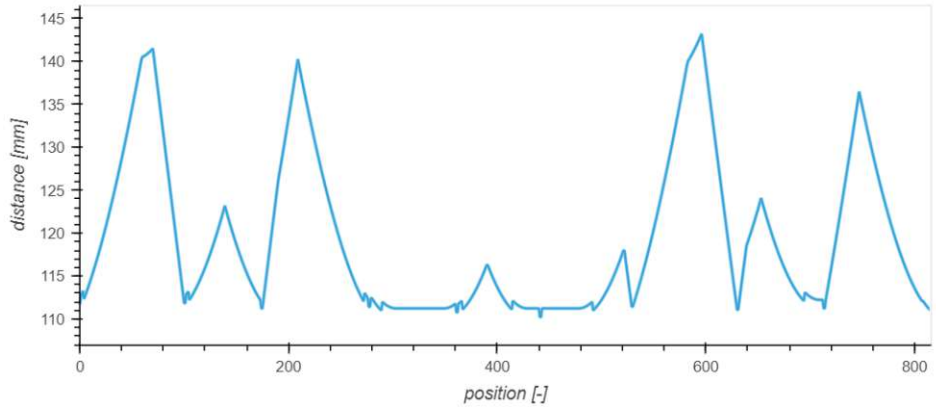

Abb. 5.26: Line plot for the whole path of slice 0 of the plexiglass structure.

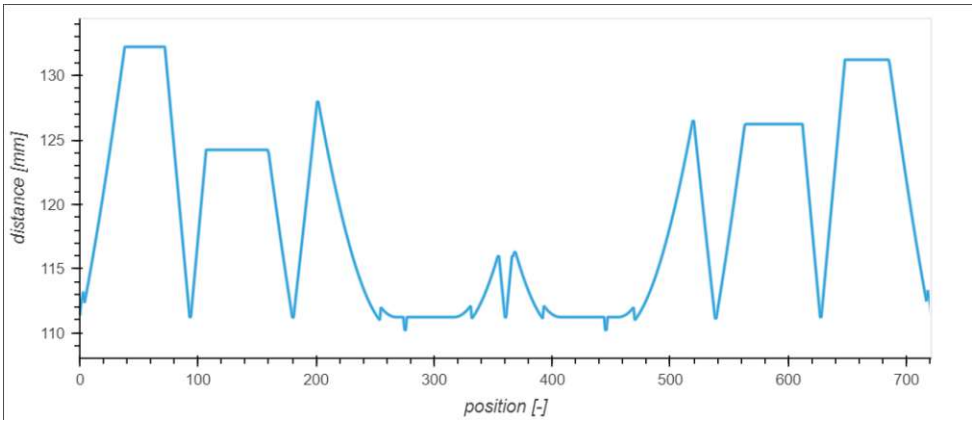

Abb. 5.27: Line plot for the whole path of slice 1 of the plexiglass structure.

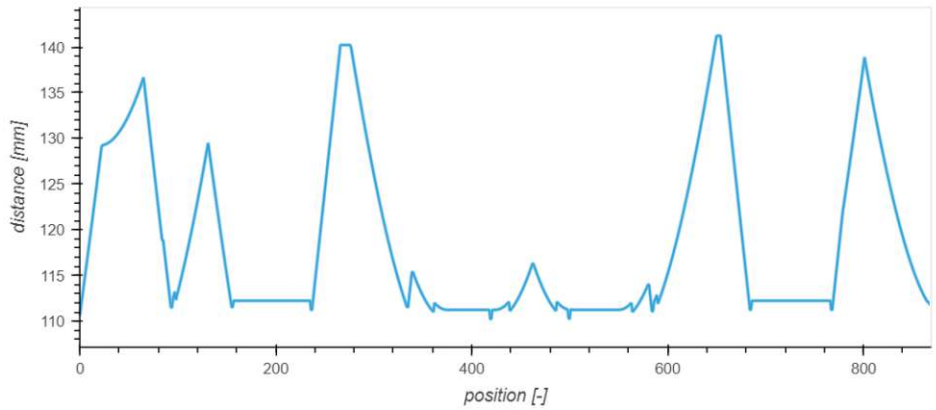

Abb. 5.28: Line plot for the whole path of slice 2 of the plexiglass structure.

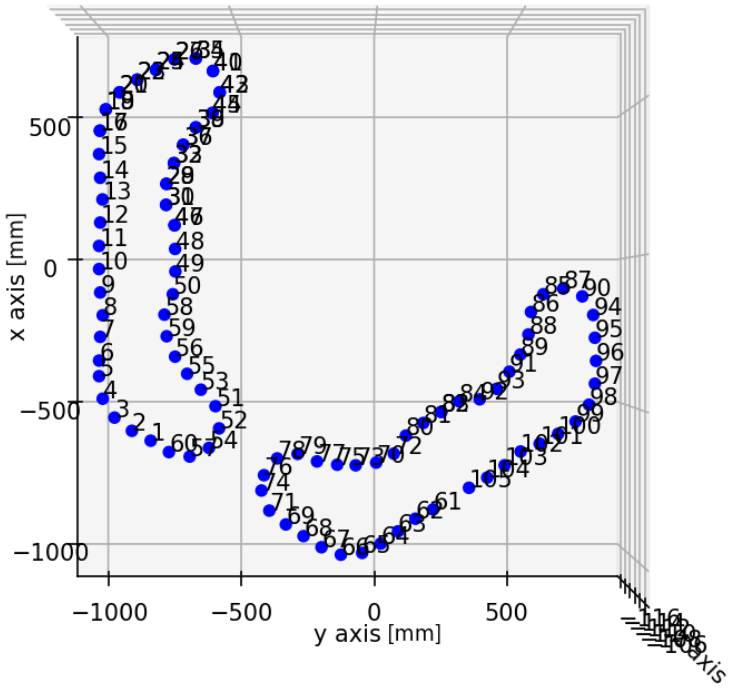

Abb. 5.29: Visualisation using the CR_plotter for slice 0 of the plexiglass structure in -z direction onto the xy-plane. Note, how the base of the structure consists of legs. Due to the algorithm settings one leg was considered to be outside. The panel in question sits to the left.

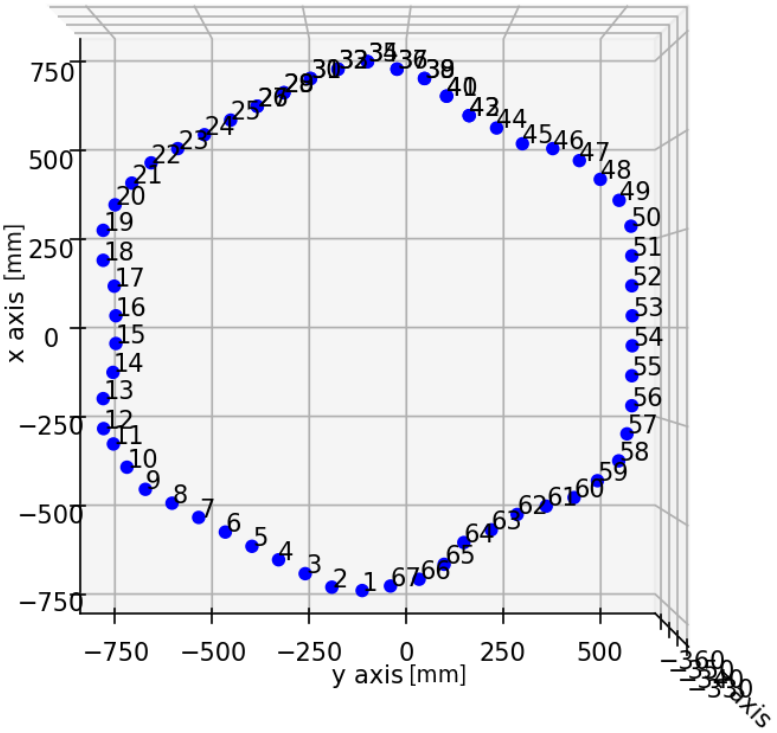

Abb. 5.30: Visualisation using the CR_plotter for slice 2 of the plexiglass structure in -z direction onto the xy-plane.

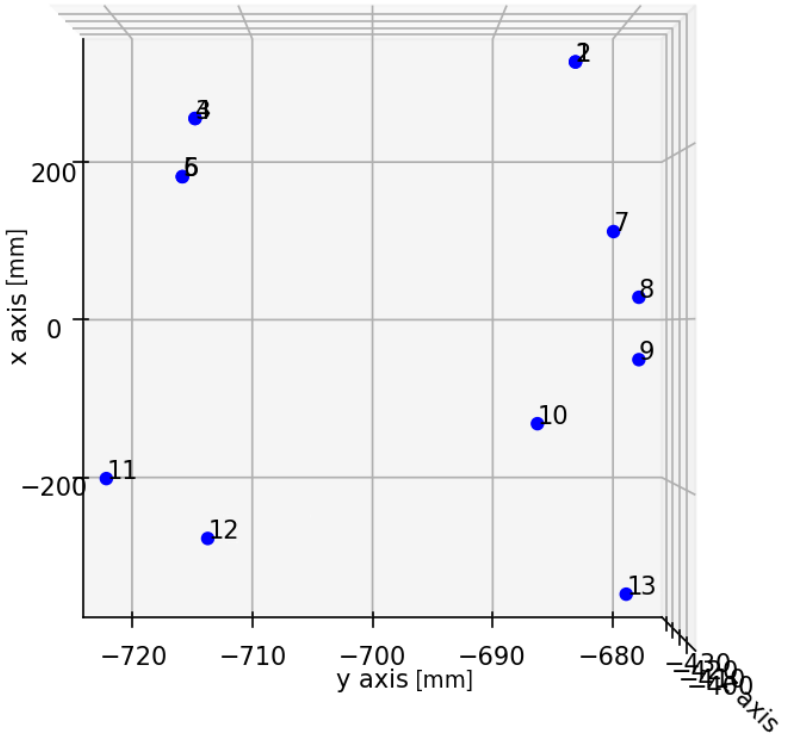

Abb. 5.31: Visualisation using the CR_plotter for slice 0 of the plexiglass structure in -z direction onto the xy-plane. The cleaning points were selected from Figure 5.29

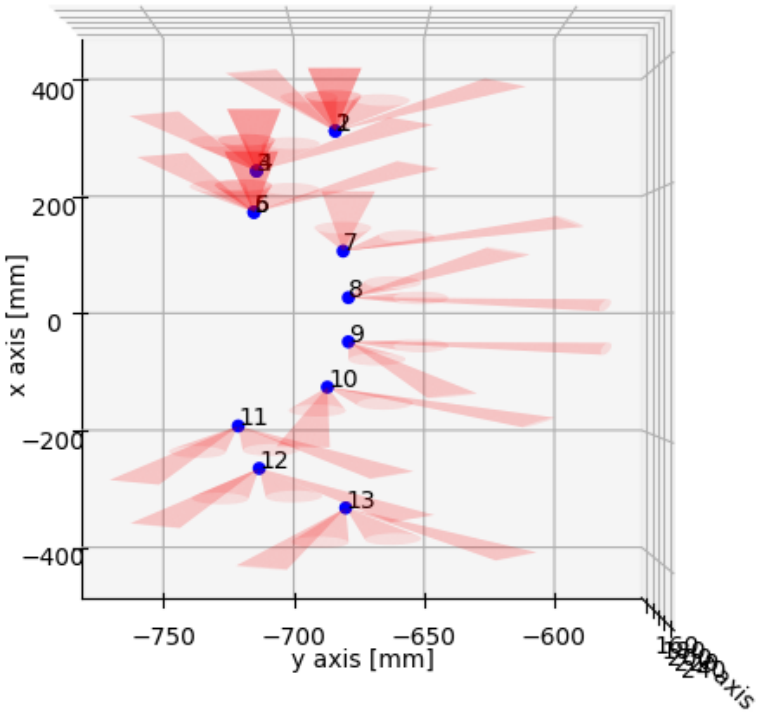

Abb. 5.32: Visualisation using the CR_plotter with cones for slice 0 of the plexiglass structure in -z direction onto the xy-plane. Note, that points 1, 2 and 13 were omitted in the final run.

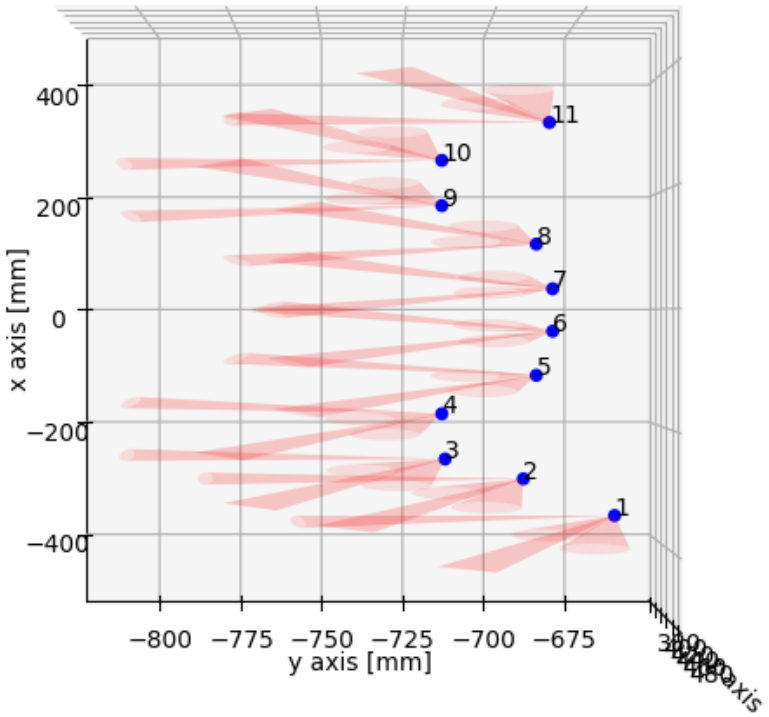

Abb. 5.33: Visualisation using the CR_plotter with cones for slice 2 of the plexiglass structure in -z direction onto the xy-plane.

5.2 Point Extraction

From the shown images it can be seen that the cleaning points are distributed over the inside surfaces of the geometries. For all geometries, with the exception of the funnel, the safety distance is kept at all times to all geometries with the given settings (REQ1). When verifying the algorithm for its correctness, it was realized that findContours from OpenCV places the found contour points on the black side of a binary image, see Figure 4.8. As a consequence, the points for cleaning are placed inside the area of the safety distance, which is not allowed. To compensate for this, 1 mm is added internally to the safety distance, since this is the maximum distance between grid points. For the cuboids, where 100 mm is the resulting constant distance for the cleaning points, the grid points sit exactly at the contours.
With regards to the funnel, where points were located lower than the required safety distance, one missing shadowing was found to be the cause. Using the single_collision_point tool, see Figure 5.34 it is apparent, that a custom height at the bottom of the geometry was necessary to account for the smaller diameter. A custom height at 0.01 mm was used, 0.0 mm does not create any slices. This behaviour was observed in previous versions and is attributed to Open3D. Figures 5.35 to 5.37 show the results for using the custom height with satisfactory results and shows how important the correct shadowing and custom heights can be.

Looking at the CR plots it is obvious, that there is room for improvement. The cones from different cleaning points do not overlap in any of the geometries. A change in settings, possibly different for various geometries and settings, or a different approach are necessary to improve on this topic.
Regarding the plexiglass models, it was observed, that the distance varies much more

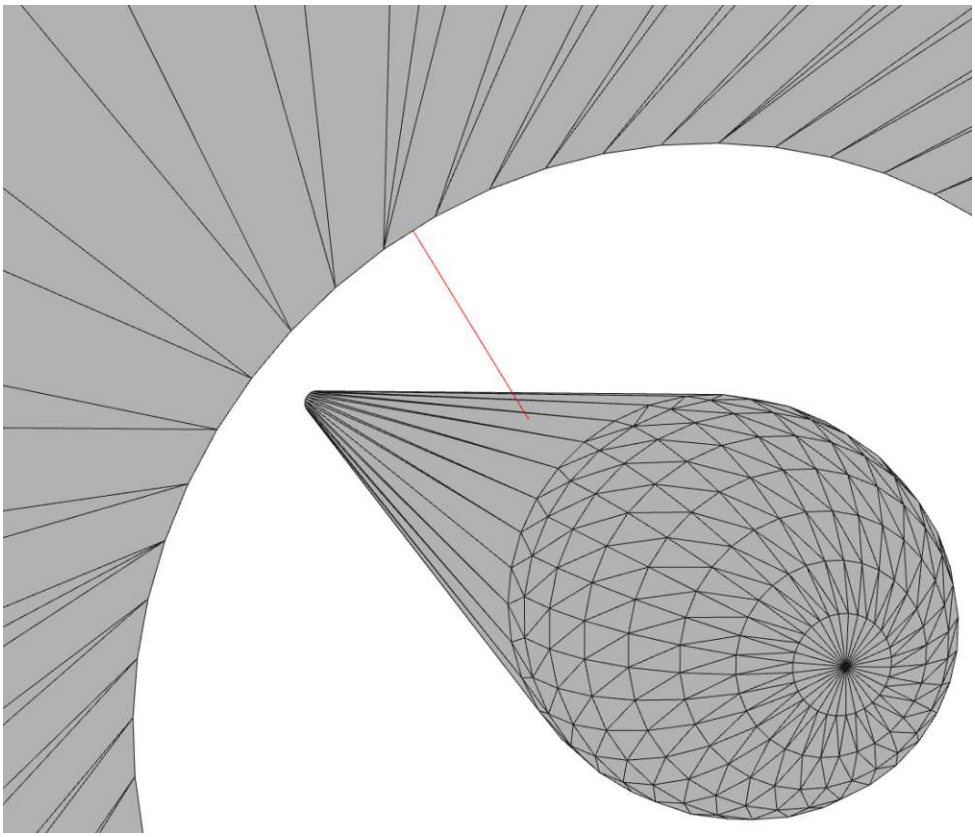

Abb. 5.34: Application of the single_collision_point tool to a cleaning point in the point recipe of the funnel, where the safety distance is not kept. Visible are the funnel, the lance, as well as a thin red line indicating the line of shortest distance. Note, how the red line does not originate from the lance head. From this image it can be concluded that missing shadowing is the cause for not keeping the safety distance.

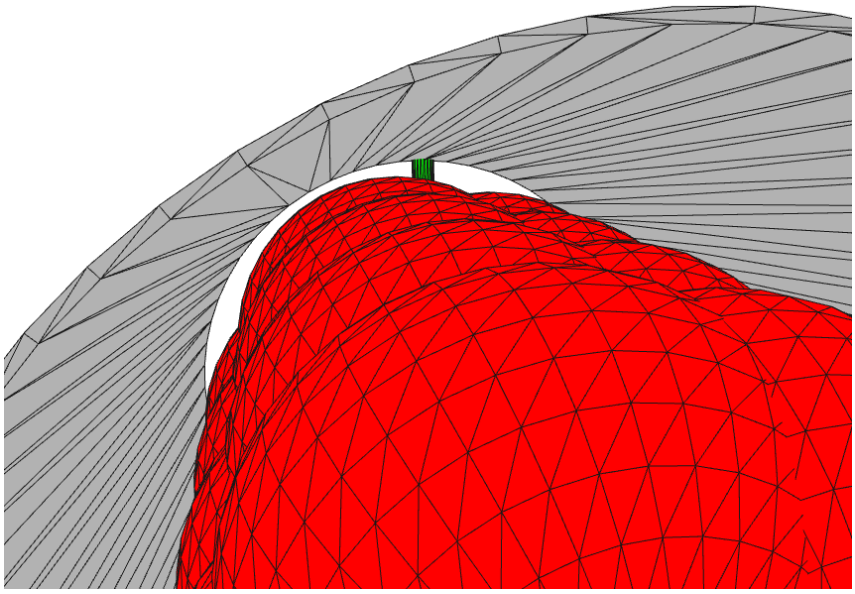

Abb. 5.35: Detailed visualisation of the extracted cleaning points of the funnel with the use of the custom height. Note, how, in comparison to Figure 5.16, there is clearly a gap visible.

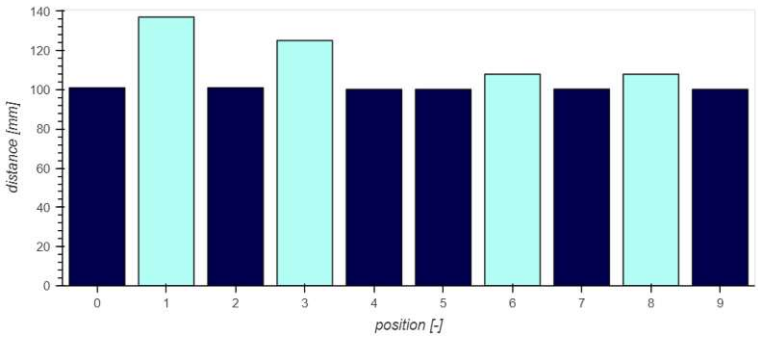

Abb. 5.36: Barplot of the cleaning points and the corner points of the funnel with the use of custom height. Note, how, in comparison to Figure 5.18, the safety distance is kept at all points.

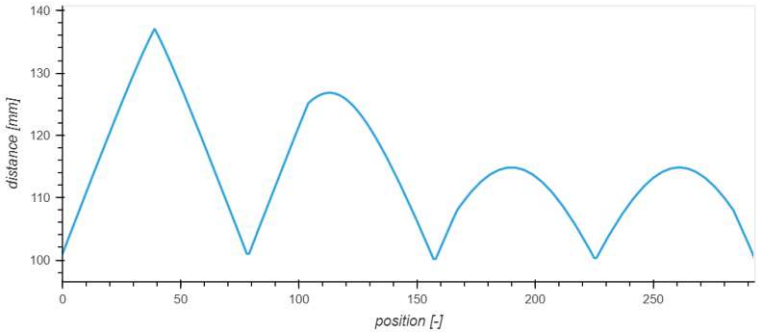

Abb. 5.37: Lineplot for the whole path of the funnel with custom height.

than for the simple geometries, see Figure 5.23 and 5.24. Yet, the safety distance is kept at all times (REQ1). The exact reason behind the larger variations in distance is unknown, but not of major concern, since the tolerance distance interval is around two and a half times larger than the observed variations in distance. One possible source for this behaviour could be the custom height. With regards to the C and R angles extracted by the algorithm, a correction by hand is necessary for heights 0 and 1. Here, the placement of the reference point leads to a failure in the algorithm as shown in Figure 4.7. From Height 2 the cones of the CO2 jet appear to be overlapping and facing the right direction.

5.3 Pathing

Looking at the pathing between cleaning points, it can be seen, that for the cuboid and the funnel only one corner point was inserted by the algorithm[2]. For the hollow pyramid there exists one occasion, where multiple points were inserted. An investigation showed, that this is the result of point 8 sitting near one of the corners of the extracted contour, making a zig-zag motion necessary.
With regards to the pathing for the hexagon structure it can be seen, that in most cases more than one corner point was necessary to get from one cleaning point to the next. In two cases even up to six points were necessary. Yet, there were some occasions where only one corner point was necessary to traverse from one cleaning point to the next.

[2]If the two points were not at the same position due to the C angle limit.

5.4 Collision Control

Concerning the collision control an overall improvement was obtained. In combination with the laser distance sensor it is now clear what the differences between the part's and the machine's coordinate systems are and what corrections have to be made to align them. Next, as was also demonstrated in several figures, the distances between the lance and the hardware can be displayed in graphs (REQ8). With regards to quality, as can be seen in Figure 5.34, not only the tip of the lance, but the whole lance can be considered in the collision control (REQ1).
Finally, the issue of different decimal points is also solved by printing all numerical values with two decimals.

Concerning the practical test with the hexagonal plexiglass structure (REQ5), it could be shown that the algorithm did indeed return a viable point recipe. No collisions were caused and the laser distance sensor did not interrupt the run.

5.5 Performance

Looking at the performance of the collision control (REQ8), it now only takes seconds to minutes to see and validate the distances for one slice. Initialising the necessary libraries takes around 13 seconds, which is not much of an issue, since, through the use of Jupyter notebooks, the loading of the libraries can be skipped, and the other cells can be run independently. Creating the shown bar plots and line plots takes fractions of a second. Loading the whole path for one slice of cuboid_1200 takes around 11 seconds, for the plexiglass structure it took around 2.5 minutes. Overall, the required time for collision control could be reduced, and, by considering the whole lance for collision control, an overall improvement in quality was achieved.
Looking at Figure 5.38 the time needed to generate the C and R angles is shown. While for lower number of points the old algorithm finishes faster, which is not noticeable for the user and stems from initialising the library, the new algorithm outperforms the old one for larger numbers of points. For example, for 10^4 points an improvement factor of around 420 could be achieved.
Looking at Figure 5.39, the performance of the main program is shown over various sizes of cuboids. What can be noticed is, that, as expected, the runtime increases with the size of the geometry. Also, the path creation accounts for most of the time needed, making up between 88.51% and 99.34% of the complete runtime. Assuming the x axis is also logarithmic, it can be seen that the runtime increases exponentially.

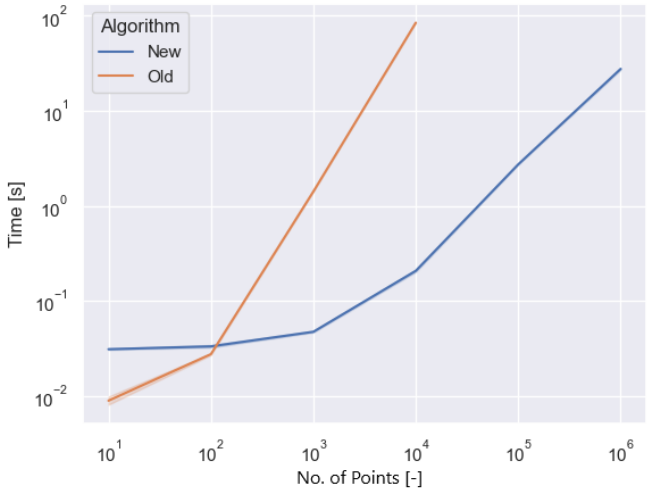

Abb. 5.38: Comparison in performance for the old and new version of the polar centre macro. Shown is the mean value of all runs and the standard deviation interval, which is barely visible. The old implementation outperforms the new one for up to 10 points. This is attributed to loading the libraries. For larger number of points the new implementation outperforms the old one. For 10^4 points a factor of around 420 lies between the two implementations. Both versions have a linear increase in execution times for larger number of points.

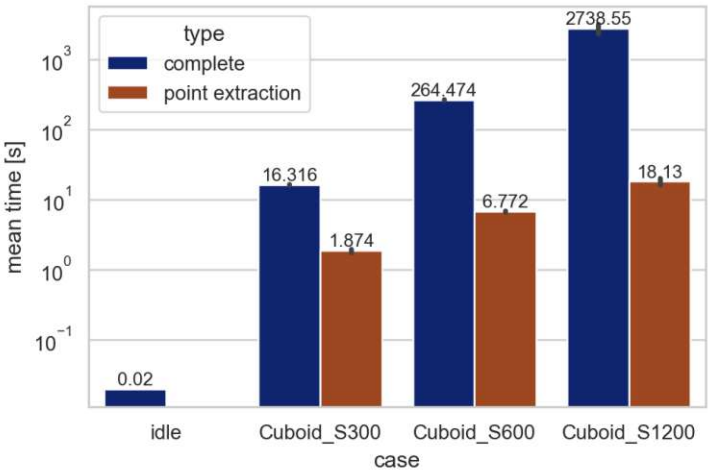

Abb. 5.39: Comparison in performance for the main program for different sizes of the cuboid. Shown are the mean values and the standard deviation. Complete refers to the total time duration, including the pathing, making the point extraction part of the complete value. The idle part of the program makes a small fraction of all runs. As expected, the runtime increases for both the point extraction as well as the total runtime. As can be seen, the point extraction also makes up a small fraction of the complete runtime, 11.49%, 2.56% and 0.66% respectively for each case.

6. Discussion

6.1 Fulfillment of Requirements

Referring back to the research question of this thesis, one approach to automate the extraction of positions for cleaning a 3D structure as well as steering the cleaning apparatus between these points was shown. A possible procedure is given, explaining how to operate the algorithm provided.
Looking at the requirements, the safety distance can be kept at all times over the whole length of the lance (REQ1), although some cases require the application of custom heights. The behaviour of the machine could be accounted for (REQ2). No diagonal movement is permitted in the pathfinding algorithm and a temporary solution for the reduction of corner points was applied to reduce the time needed for traversal. As input for the algorithm serves an stl file, which is a common CAD format (REQ3). The output from the algorithm are csv files for different heights, which contain all points necessary for operations (REQ4). A way to connect these csv files, as well as a suggested tree builder are provided. During a practical run no collisions and no interruption from the laser distance sensor occurred (REQ5). In the geometries used for validation no cleaning point was found placed further than 120 mm (REQ6). The usage of Open3D made it possible to not have to rely on CAD software (REQ7). A reduction in the time necessary to detect violations of the safety distance could be achieved, while also improving the quality of the collision control (REQ8). As could be shown, the distances can also be visualised at all positions of the lance. Finally, through the use of switches it is possible to apply the algorithm not only for inside cleaning, as was shown in this document, but also for the outside cleaning of components (REQ9).

6.2 Future Work

The development of the presented algorithm is not yet completely finished. For example, in the contour selection process another mode could be added which saves the list with the selected contours so if the algorithm is repeated the choice does not have to be repeated. When fitting the ellipse to obtain the reference point, in case there are too few points one could also take the mean of all cleaning points to further reduce the user interaction. In case the length of the contour is too small, one could also take the reference point as a cleaning point. This would lead to a complete spherical cleaning point, cleaning in all directions possible by the machine. Considering the contour is already small, this could optimise the cleaning results in narrow structures.
As was presented in this document, the C angle determination is not sufficient. In the future more possibilities to obtain or prescribe the C and R angles should be considered for different use cases. For example, when cleaning a simple wall, a constant C and R setting would already be enough. To improve the current determination of C and R angles, the use of Open3D with its boolean operations should be considered. This could reveal the regions reachable by the CO2 jet and help in extracting the necessary angles. In case no structure can be reached at all, for example when located at an orifice, the algorithm could then delete the cleaning point. Together with the saved binary images, this could be done in a robust manner. The boolean functions in Open3D could also be used to determine if all surfaces of interest are covered. By using the

available cones in the library, sections of the structure could be removed leaving surfaces which are not captured by the CO2 jet. Other forms of visualising the C and R angles using Open3D would already be a first step in validating the necessary C and R angles. Finally, the actual dimensions of the CO2 jet must be determined in an experimental setup.
When extracting the cleaning points from the contours in the algorithm it should be considered to look out for points where contours have corners. Putting cleaning points at such points may lead to a better cleaning result than with the current implementation, where cleaning points are set at equidistant length. This could use corner detecting algorithms in OpenCV and in between said corners, a similar algorithm as implemented could place points where there are too few points.
As mentioned above, it was realized that findContours from OpenCV places the found contour points on the 'wrong' side of a binary image. As a consequence, the points for cleaning are placed inside the area of the safety distance. Since a proper correction takes more than just the inversion of the images, as a temporary solution, the safety distance is increased inside the algorithm by 1 mm since that is the maximum error. As a result, the pathfinding can loose two options to move from one cleaning point to the next. For example, for the cuboids, the machine could go along the contours. But due to this displacement, traversals of 1 mm can occur to get out of the 'pit' to then move to the next point, see 4.8 and 5.11. Correcting this behaviour could lead to an improvement in pathfinding and improve the correctness of the algorithm.
Another point for the future is the pathfinding itself. While for the viewed cases the current implementation was sufficient, two major issues can be extracted for future development:

1. A more robust solution to reduce the number of corners in the paths is recommended. One suggestion is a new definition of the heuristic function of the A* algorithm. It could penalise performing turns or include some form of momentum while still searching for the shortest path.

2. The pathfinding is responsible for most of the execution time. A closer look into which parts of the algorithm are the cause for this is recommended. It is expected that the many read and write operations are the cause but the influence of the A* algorithm should also be investigated. Parallelisation could also serve as a possible solution to accelerate pathfinding.

To obtain a complete point recipe automatically, some more steps have to be taken. First of all, the temporary script tree_creation serves as a possible input as to which contour of the part of interest is connected to which. For plotting the tree it is recommended to use NetworkX in combination with hvplot since it offers an interactive display of the connections between nodes. After building a complete tree, a depth first traversal with postordering is suggested as a cleaning sequence.
Also, when obtaining a complete point recipe, the very first cleaning point must be preceded by a point which lies at $z = 0$ with the x and y coordinates of the first cleaning point to avoid collision of the machine with the frame carrying the hardware. Subsequently, when changing heights in the point recipe, a point must be inserted before the first cleaning point with the x and y coordinates of the previous point but at the height of the next point. This is recommended to be done before the pathing is done to obtain possible necessary corner points in the point recipe and the whole path to this first cleaning point, leading to better collision control.
Concerning collision control itself it is suggested to apply the available widgets available

in hvplot. Sliders could be used to facilitate the change between different slices. Also, the insertion of custom heights could be automated, resulting in an iterative scheme, leading to a higher degree of automation.
To better understand what the algorithm is doing and where it failed, logging is recommended. For better documentation it is advised to use libraries such as Sphinx to create a complete documentation from the docstrings created. Also, to be able to save the used settings of different parts, is is recommended to create and use init files.
To improve the application of the algorithm a GUI or web browser interface is recommended to make the tool accessible to people with no coding experience.

In future work it is recommended to show the cleaning efficiency of the algorithm in a practical manner. A possible setup configuration is documented with images as well as a procedure.
Concerning tomography, it could be possible to measure the time needed for particles to travel along the lance into the suction device. Relating real-time position and angle of the lance could relate particles and cleaning status of the hardware. Visualisation in Open3D could then help focusing the cleaning efforts in the areas where it is needed.

7. Summary

In the presented work an overview and the importance of cleanliness in the context of spaceflight was presented. A focus was put onto particulate contamination, the application of CO2 snow cleaning and how it is currently done at OHB. Recently, larger satellite components, such as baffles, should be cleaned by this method, requiring a machine to perform the procedure.

Based on the V-model, the system was first presented. Derived from the description problems, objectives and requirements were derived. They formed the basis for the concept and programming choices. The second part explains the implementation of an algorithm which must guarantee that, given a stl file of a structure, a given safety distance of the lance is kept when cleaning from the inside. The limits of the algorithm as well as how the issue of pathing between cleaning points is solved and various tools to verify the distance are presented. A suggested process and performance measurements finalize the implementation of this work.

The verification as the third part is presented and discussed using the different available tools. This is done by applying the algorithm to different geometries as well as a hexagonal plexiglass structure. As a main result it could be shown that, generally speaking, the algorithm performs as expected. The safety distance is kept at all times and this can be verified along the whole lance. Yet, the distances for the hexagonal structure vary much more than for the other geometries. In a practical run with the plexiglass structure no collision was caused and the laser distance sensor in the lance did not interrupt the run. Finally, the C and R angles still require quite some interaction with the user and the pathing needs further attention.

The presented work can serve as a basis for future development. Corrections in the algorithm, such as the placement of the contour points on the wrong side, further automatisation as well as new concepts, e.g. to obtain the necessary angles for operations or improved pathing can contribute to a higher degree of automation and better cleaning results.

Bibliography

[1] ECSS, "Ecss-q-st-70-01c – cleanliness and contamination control," European Space Agency, Tech. Rep., 2008.

[2] L. Gail, U. Gommel, and H.-P. Hortig, *Reinraumtechnik*. Springer, 2012.

[3] Fraunhofer Institute, *Technical cleanliness*, 2021. [Online]. Available: https://www.ipa.fraunhofer.de/en/expertise/ultraclean-technology-and-micromanufacturing/cleanliness-technology/technical-cleanliness-in-the-automotive-industry.html.

[4] A. C. Tribble, *Fundamentals of contamination control*. SPIE Press, 2000, vol. 44.

[5] R. Kohli and K. L. Mittal, *Developments in Surface Contamination and Cleaning: Fundamentals and Applied Aspects*. William Andrew, 2008.

[6] D. Bolmatov, V. V. Brazhkin, and K. Trachenko, "Thermodynamic behaviour of supercritical matter," *Nature communications*, vol. 4, no. 1, 2013. DOI: https://doi.org/10.1038/ncomms3331.

[7] S. P. Sawan, *Supercritical fluid cleaning: fundamentals, technology and applications*. Noyes Publications, 1998.

[8] UN Environment Programme, *The montreal protocol*, 2018. [Online]. Available: https://www.unep.org/ozonaction/who-we-are/about-montreal-protocol.

[9] R. B. Gupta and J.-J. Shim, *Solubility in supercritical carbon dioxide*. CRC press, 2007.

[10] acp systems AG, *Was ist co2-schneestrahlreinigung?* 2023. [Online]. Available: https://www.acp-systems.com/co2-schneestrahlreinigung/.

[11] D. T. Hoppe, "Automated carbon dioxide cleaning system," ser. NASA, Washington, Technology 2001: The Second National Technology Transfer Conference and Exposition, Volume 2, 1991.

[12] European Space Agency (ESA), *Cleaning the herschel primary mirror*, 2019. [Online]. Available: https://sci.esa.int/web/herschel/-/44315-cleaning-the-herschel-primary-mirror.

[13] National and Aeronautics Space Administration (NASA), *Engineers clean mirror with carbon dioxide snow*, 2015. [Online]. Available: https://www.nasa.gov/image-article/engineers-clean-mirror-with-carbon-dioxide-snow/.

[14] Z. A. Algfoor, M. S. Sunar, and H. Kolivand, "A comprehensive study on pathfinding techniques for robotics and video games," *International Journal of Computer Games Technology*, vol. 2015, 2015. DOI: https://doi.org/10.1155/2015/736138.

[15] D. Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah, and E. Gunawan, "A systematic literature review of a* pathfinding," *Procedia Computer Science*, vol. 179, pp. 507–514, 2021. DOI: 10.1016/j.procs.2021.01.034.

[16] J. Ichnowski, Y. Avigal, V. Satish, and K. Goldberg, "Deep learning can accelerate grasp-optimized motion planning," *Science Robotics*, vol. 5, no. 48, eabd7710, 2020. DOI: 10.1126/scirobotics.abd7710.

[17] acp systems AG, *The bmw works in landshut introduces co2 snow jet cleaning before painting plastic components*, 2010. [Online]. Available: https://www.pressebox.com/pressrelease/acp-advanced-clean/The-BMW-works-in-Landshut-introduces-CO2-snow-jet-cleaning-before-painting-plastic-components/boxid/317333.

[18] acp systems AG, *Complex robot applications simply implemented*, 2022. [Online]. Available: https://www.pressebox.com/pressrelease/acp-advanced-clean/complex-robot-applications-simply-implemented/boxid/1096145.

[19] ArtiMinds, *Offline-programmierung simulation mit artiminds rps*, 2023. [Online]. Available: https://www.artiminds.com/robotics-software-and-services/offline-robot-programming/.

[20] M. Lunny, "Automation of nc programming with artificial intelligence," Ph.D. dissertation, Massachusetts Institute of Technology, 2022.

[21] C.-C. Ho, *Feature-based process planning and automatic numerical control part programming*. The University of Utah, 1997.

[22] *Vdi/vde 2206 "entwicklung mechatronischer und cyber-physischer systeme"*, 2019. [Online]. Available: https://www.vdi.de/richtlinien/programme-zu-vdi-richtlinien/vdi-2206.

[23] A. Bresser, *Python-pathfinding*, 2023. [Online]. Available: https://github.com/brean/python-pathfinding.

[24] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[25] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[26] *Hvplot homepage*, 2024. [Online]. Available: https://hvplot.holoviz.org/index.html.

[27] *What is three-tier architecture?* 2024. [Online]. Available: https://www.ibm.com/topics/three-tier-architecture#:~:text=Three%2Dtier%20architecture%20is%20a,associated%20with%20the%20application%20is.

[28] *Permuted congruential generator (64-bit, pcg64)*, 2022. [Online]. Available: https://numpy.org/doc/stable/reference/random/bit_generators/pcg64.html#numpy.random.PCG64.

[29] *Bit generators*, 2022. [Online]. Available: https://numpy.org/doc/stable/reference/random/bit_generators/index.html#random-bit-generators.

List of Figures

List of Tables