Virtual Reality-based CMM Operation Training and Efficient Probe Path Planning Using A* and Heuristic Methods

By

Uzair Khan

A Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
of the University of Manitoba
in partial fulfilment of the requirements of the degree of

Master of Science

Department of Mechanical Engineering
University of Manitoba
Winnipeg

Abstract

This thesis presents the design and implementation of a virtual reality (VR)-based training and path-planning system for Coordinate Measuring Machine (CMM) operations. With the increasing integration of automation and digital technologies in manufacturing, there is a growing demand for efficient, accessible, and scalable training platforms that replicate real-world inspection tasks while minimizing costs, risks, and limitations associated with physical equipment. To address this need, a virtual CMM environment is developed using Unity as a simulation engine. The system emulates CMM functionality with interactive features including the stylus qualification, coordinate system selection, dimensional measurement of geometric features, and feedback on probe movements and errors. Users gain hands-on experience by wearing a headset to learn complex metrology workflows and by navigating the environment using controllers.

The thesis also introduces a path-planning method to optimize the CMM inspection routines. Traditional CMM programming methods often rely on manual sequencing, resulting in suboptimal probe travel paths and inconsistent operations, especially when dealing with intricate part geometries. To address these challenges, a hybrid planning strategy is proposed that combines intuitive heuristic guidance with the A* (A-star) search algorithm to search for the shortest collision-free path between measurement features. The method also incorporates geometric constraints, clearance verification, and support for feature-based prioritization, enabling it to adapt to a range of inspection scenarios with varying complexity.

The developed system is validated through a series of simulated measurement tasks on representative 3D models, comparing the optimized path-planning results with manually defined conventional sequences. The automated planning approach achieved an 8.4% reduction in total probe travel distance compared to a manual baseline, with planning computation taking under 1 second. Qualitative and quantitative feedback from user evaluations (n = 30) conducted using the System Usability Scale further confirms the system's usability with a mean SUS score of 76.4, significantly above the 68-point benchmark, highlighting its potential as a supplementary training resource in educational and industrial contexts. By unifying immersive VR-based learning and intelligent path planning into a single platform, this research contributes to the broader objectives of smart manufacturing, digital twin development, and next-generation metrology education. The

proposed solution not only enhances operator readiness but also lays the groundwork for automated and adaptive inspection strategies in manufacturing systems.

Keywords

Virtual Reality, Path Planning, Coordinate Measuring Machine, Inspection Planning, Machine Operation Training

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Qingjin Peng, for his invaluable guidance, unwavering support, and patience throughout my master's research. His expertise in virtual manufacturing and design provided the intellectual foundation for this work, and his mentorship has been instrumental in shaping both my research capabilities and professional development. I am truly grateful for the trust he placed in me and for the countless hours he dedicated to reviewing my work and steering it in the right direction.

I gratefully acknowledge the financial support provided by Mitacs through the Lab2Market Validate Program and the fellowship by Prof. Peng, which not only funded a significant portion of my research but also offered me a unique opportunity to explore the commercialization potential of my work. This experience broadened my perspective on translating academic research into real-world impact.

I am thankful to my colleagues in the Virtual Manufacturing Laboratory, Waqar Shehbaz and Waseem Ahmed, for the stimulating discussions, mutual encouragement, and camaraderie that made the long hours in the lab far more enjoyable. A special thank you goes to the faculty and staff of the Department of Mechanical Engineering at the University of Manitoba for fostering a supportive and enriching academic environment.

Above all, I owe an immeasurable debt of gratitude to my family. To my parents whose sacrifices, unconditional love, and constant prayers have been the bedrock of every achievement in my life. To my sister, Rabia, for always believing in me and being a source of strength and motivation. This thesis is as much yours as it is mine.

Finally, I am grateful to the Almighty for bestowing upon me the strength, perseverance, and clarity of mind needed to complete this work. Every step of this journey has been a reminder of the blessings I am surrounded by.

Publications

Conference paper:

Khan, U. K., & Peng, Q. (2024). Virtual reality-based system for coordinate measuring machine operation training. In Proceedings of the CAD'24, Eger, Hungary, July 8–10, 2024, pp. 351–355. https://doi.org/10.14733/cadconfP.2024.351-355

Ahmed, W., Khan, U., & Peng, Q. (2025). Augmented reality-based operation training for coordinate measuring machines using user-centered interface approach. In Proceedings of the CAD'25, Shenzhen, China, May 23–25, 2025, pp. 272–277. https://doi.org/10.14733/cadconfP.2025.272-277

Journal article:

Khan, U. K., & Peng, Q. (2025). Virtual reality-based training system of coordinate measuring machine operations. Computer-Aided Design & Applications, 22(5), 867–881. https://doi.org/10.14733/cadaps.2025.867-881

Ahmed, W., Khan, U., & Peng, Q. (2025). Augmented reality-based operation training for coordinate measuring machines using user-centered interface approach. Computer-Aided Design & Applications. https://doi.org/10.14733/cadaps.2027.1-18

List of Tables

List of Figures

List of Abbreviations/Symbols

Technology & Systems

VR	Virtual Reality
AR	Augmented Reality
XR	Extended Reality (umbrella term for VR, AR, and MR)
CMM	Coordinate Measuring Machine
VCMM	Virtual Coordinate Measuring Machine
HMD	Head-Mounted Display
CAD	Computer-Aided Design
GUI	Graphical User Interface

Standards & Platforms

OpenXR	Cross-platform XR runtime API standard (Khronos Group)
XRIT	XR Interaction Toolkit (Unity framework for VR interactions)
URP	Universal Render Pipeline (Unity's cross-platform graphics pipeline)
DMIS	Dimensional Measuring Interface Specification (CMM programming standard)
EUKOM	Europäisches Kompetenzzentrum für Koordinatenmesstechnik
AUKOM	Ausbildung Koordinatenmesstechnik (CMM training certification program)
GD&T	Geometric Dimensioning and Tolerancing

Coordinate Systems

MCS	Machine Coordinate System (CMM internal reference frame)
PCS	Part Coordinate System (workpiece-aligned reference frame)

Algorithms & Optimisation

TSP	Travelling Salesman Problem
GTSP	Generalized Travelling Salesman Problem
NN	Nearest-Neighbour heuristic (greedy TSP construction)
2-opt	Two-edge exchange local improvement heuristic for TSP tours

A*	A-Star heuristic search algorithm for shortest collision-free path
GA	Genetic Algorithm
ACO	Ant Colony Optimization
PSO	Particle Swarm Optimization
RL	Reinforcement Learning
AABB	Axis-Aligned Bounding Box (collision detection bounding volume)
IoU	Intersection over Union (duplicate contour removal metric)
CCW	Counter-Clockwise (orientation test in line-segment intersection)

Statistics & Evaluation

RMSE	Root Mean Square Error
LOA	Limits of Agreement (Bland–Altman ±1.96 s_d interval)
SUS	System Usability Scale (10-item questionnaire, score 0–100)

Key Mathematical Symbols

τ	Feature visitation sequence (permutation / TSP tour)
$L(\tau)$	Total path length for tour τ: $\Sigma\, d(c_{\tau_i}, c_{\tau_i+1})$
d(p, q)	Euclidean distance between points p and q
f(w)	A* total cost: f = g(w) + h(w)
g(w)	A* actual travel cost from start to node w
h(w)	A* heuristic (Euclidean distance from w to goal)
s	Scale factor (pixels per inch)
D_{ref}	Reference (nominal) dimension from CAD
D_{VR}	Dimension measured in the VR system
D_{CMM}	Dimension measured on the physical CMM
$\bar{e}$	Mean measurement error (bias) across N measurements
σ	Standard deviation of measurement errors
$\bar{d}$	Mean difference between VR and CMM measurements (Bland–Altman)
α	Cronbach's alpha (internal consistency of SUS questionnaire)
R^2	Coefficient of determination (VR vs. CMM correlation)
n	Number of features/targets in path planning; or participants (n = 30)

Table of Contents

Chapter 1: Introduction

1.1 Coordinate Measuring Machines

A coordinate measuring machine (CMM) is a precision instrument to measure the geometry of physical parts by sensing discrete points on their surfaces with a probe (as shown in Figure 1.1(a)). In a typical CMM operation, the probe (which can be mechanical touch-trigger, optical, or laser) moves along three orthogonal axes (X, Y, and Z) to capture the coordinates of points on the workpiece's surface. Each contact or detection generates an (X, Y, Z) data point in a Cartesian coordinate system referenced to the machine's datum. By collecting many such points, the CMM reconstructs the part's dimensional features, enabling precise measurements of size, position, and shape. CMMs can achieve sub-micrometre measurement accuracy, making them indispensable for quality assurance in precision manufacturing.

CMMs are manufactured in several structural configurations to accommodate varying inspection requirements and part sizes. The most widely used is the bridge-type CMM, shown in Figure 1.1(a), which consists of a rigid granite base and a movable bridge that carries the probe across the workspace. The bridge architecture provides excellent structural stability and measurement precision, making it well-suited for medium-sized components. Prior to measurement, the probe is calibrated using a reference artifact. Figure 1.1(b) illustrates a calibration sphere used to determine stylus offsets and verify probing accuracy, thereby minimizing measurement errors caused by stylus geometry and mounting variations. A sample workpiece containing multiple geometric features is shown in Figure 1.1(c), including circular holes, slots, conical surfaces, and planar regions. CMM probing systems support various stylus configurations depending on the inspection requirements, as shown in Figure 1.1(d). Proper stylus selection is critical to ensure accurate measurements and to avoid collisions during probing. The specific hardware specifications and accuracy parameters of the CMM used in this study are documented in Section 5.2.

CMMs play a critical role in modern manufacturing quality assurance. They are widely used as standard equipment in inspection labs and production lines to verify that machined parts and assemblies meet design tolerances. By comparing measured coordinates and geometries to CAD

model or drawing requirements, such as dimensions, flatness, roundness, and true position, CMMs help ensure that each part conforms to specifications. In industries such as aerospace, automotive, and precision manufacturing, CMMs serve as the metrological benchmark, often for final inspection and certification of critical components. Given their ability to provide detailed quantitative feedback, CMMs also support process improvement by identifying systematic manufacturing deviations. In summary, CMMs are central to manufacturing metrology because they link a product design to its physical realization, enabling rigorous quality verification. However, exploiting a CMM's full capabilities requires skilled human operators. Operating a CMM involves not only understanding its hardware and software but also knowledge of measurement strategies, such as properly aligning a part, selecting features to measure, determining the measurement order, calibrating and orienting the probe, and avoiding collisions. The workflow for operating the CMM is presented in Figure 1.2. Mastery of CMM operations is typically achieved through practical training and experience.

(a) (b) (c) (d)

Figure 1.1: (a) Bridge CMM, (b) Calibration Sphere, (c) Workpiece, (d) Probing Type

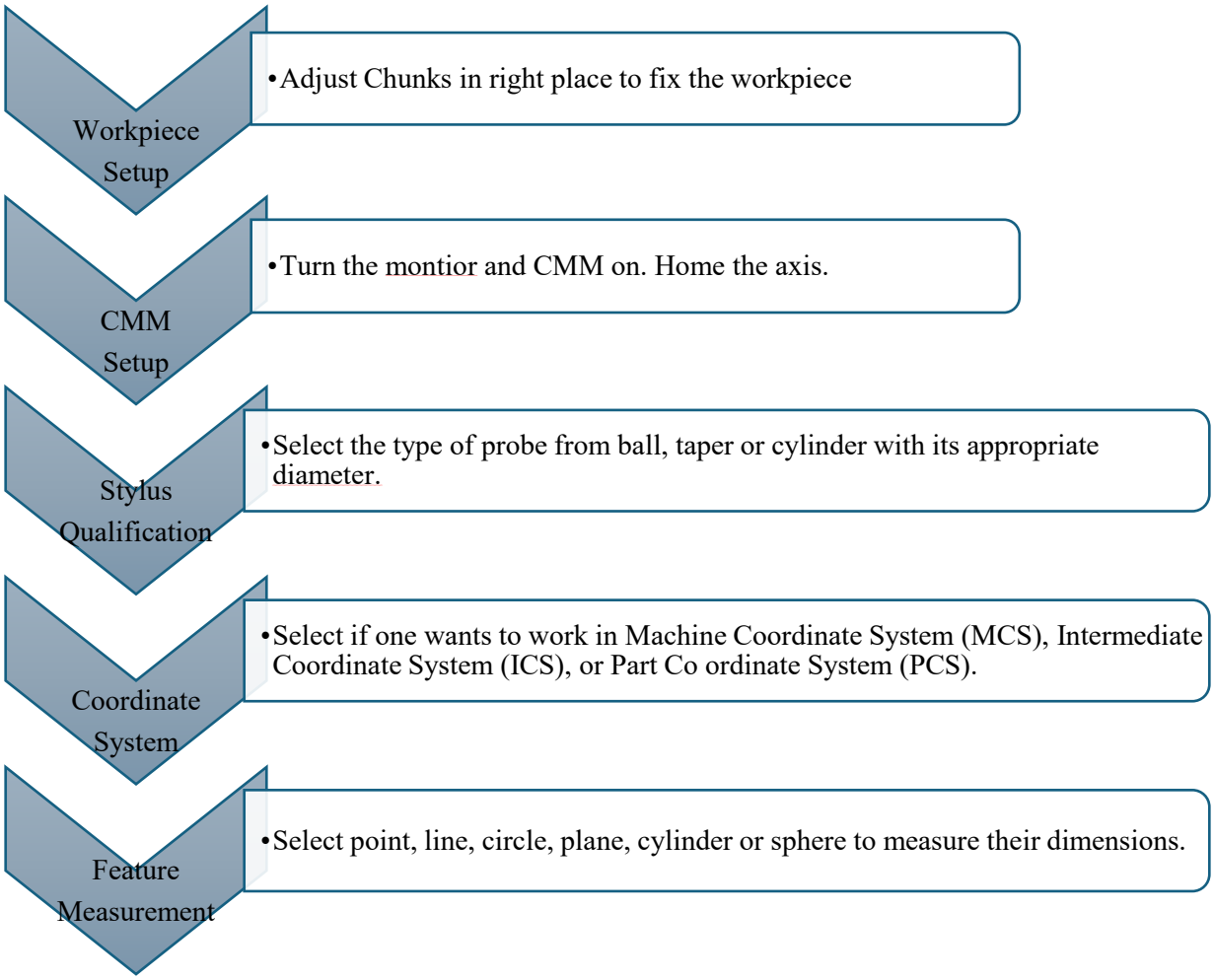


Figure 1.2: A typical CMM inspection setup and data flow

1.2 Challenges in CMM Operation Training

Conventional CMM operation training usually involves classroom theory combined with hands-on practice on an actual machine. While real-machine experience is invaluable, it is often constrained by resource availability and cost. Training institutions or companies may have limited CMMs, which can restrict the frequency and depth of practical training exercises.

Another challenge is the risk of equipment damage or probe collisions during training. CMMs and their probes are delicate and costly; a mistake by an inexperienced user (such as driving the probe into a part or fixture) can result in expensive damage or downtime. Moreover, novices might feel anxiety when operating the real machine, knowing that errors have real consequences. Therefore,

traditional operation training on actual CMMs is constrained by limited machine availability, high costs, and inherent risks.

Traditional CMM operation training is also time-consuming and instruction-dependent. Setting up a CMM for a training session (calibrating probes, fixturing parts, etc.) takes significant time. Instructors must closely supervise initial sessions to ensure safety and correctness. Complex tasks (such as multi-axis probe head use or intricate part programs) may be demonstrated rather than operated by learners due to time constraints. There is also typically a steep learning curve: trainees must familiarize themselves with both hardware operation and the metrology software interface, which can be overwhelming without iterative practice. All these factors, scarce access, high stakes, and long learning curves, show a need for improved methods to train CMM operators.

1.3 Virtual Reality–based Training Solutions for CMM Operations

Virtual reality (VR) is an immersive simulation medium that places the user inside an interactive, computer-generated 3D environment for first-person experiential learning. In mechanical engineering and related disciplines, VR, AR (Augmented Reality), MR (Mixed Reality), and desktop VR form a spectrum of "extended reality" tools that can support different training needs: immersive VR best replicates complex or hazardous scenarios; AR and MR overlay guidance and visualizations on real equipment; and desktop VR offers accessible, lower-immersion 3D training on standard PCs, where physics, collisions, and scripted interactions enable trainees to inspect machinery, rehearse procedures, and practice decision-making safely and repeatedly.

A well-rounded VR training environment is organized around four core pillars: geometry, physics, behavior, and rules, as shown in Figure 1.3 and discussed in detail in Section 2.1.4. A typical VR training pipeline combines 3D modeling tools (e.g., AutoCAD, SolidWorks, Blender), a real-time game engine (Unity, Unreal), device SDKs (OpenXR, vendor plugins), and suitable hardware (PC-based or standalone headsets with controllers and optional haptics) to deliver responsive, low-latency simulations. PC-based devices (such as high-end headsets with precise tracking) are generally favored when high visual fidelity or strict accuracy is required, while standalone headsets are preferred for mobility, scalability, and ease of graphics performance deployment. Across these platforms, empirical studies report that well-designed VR training can match or exceed traditional

methods in knowledge acquisition, procedural skill development, and safety outcomes, underscoring its growing role as an effective, scalable tool for engineering education and industrial training.

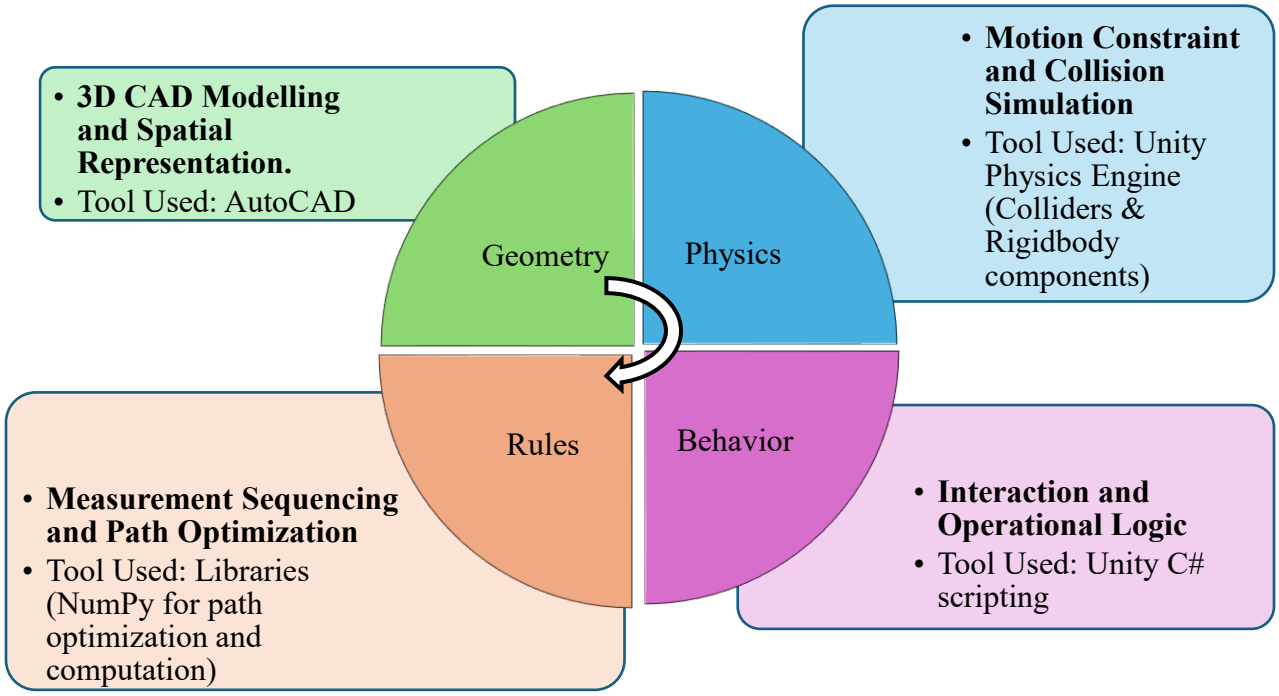


Figure 1.3: Four Pillars of the VR model

Unity is used in this study as the main platform for developing a virtual CMM system. Unity provides realistic 3D graphics, a built-in physics engine, and strong support for VR hardware. Using Unity, the 3D CAD models of the CMM and workpiece can be imported directly and combined with scripts that control the machine motion, collision detection, and user interactions. Unity also supports real-time feedback, UI overlays, and feature highlighting, making the virtual environment suitable for training and simulation. A key advantage is Unity's multi-platform deployment: the same project can run on desktop PCs, VR headsets, mobile devices, and even the Web, making the training system highly scalable. Its large asset library and community resources further accelerate development (Unity: Develop, Deploy, and Grow | The World's Leading Game Engine, 2026).

In the Unity interface, four main windows support the creation of the virtual CMM: the Hierarchy (organizes all CMM components), Scene View (3D workspace for arranging the model), Game View (shows how the simulation behaves at runtime), and Inspector (where motion parameters, scripts, and settings are adjusted), as shown in Figure 1.4. These tools collectively enable the integration of 3D models, user input, and simulation logic into a functional virtual CMM environment.

In summary, VR provides a safe, cost-effective, and highly flexible approach to CMM training. By immersing trainees in a realistic simulation built with tools like Unity, the accessibility and risk barriers of traditional training can be overcome. Trainees can gain hands-on experience operating a CMM, plan measurement sequences, and respond to machine feedback, all within a virtual environment that mirrors real-world physics and constraints.

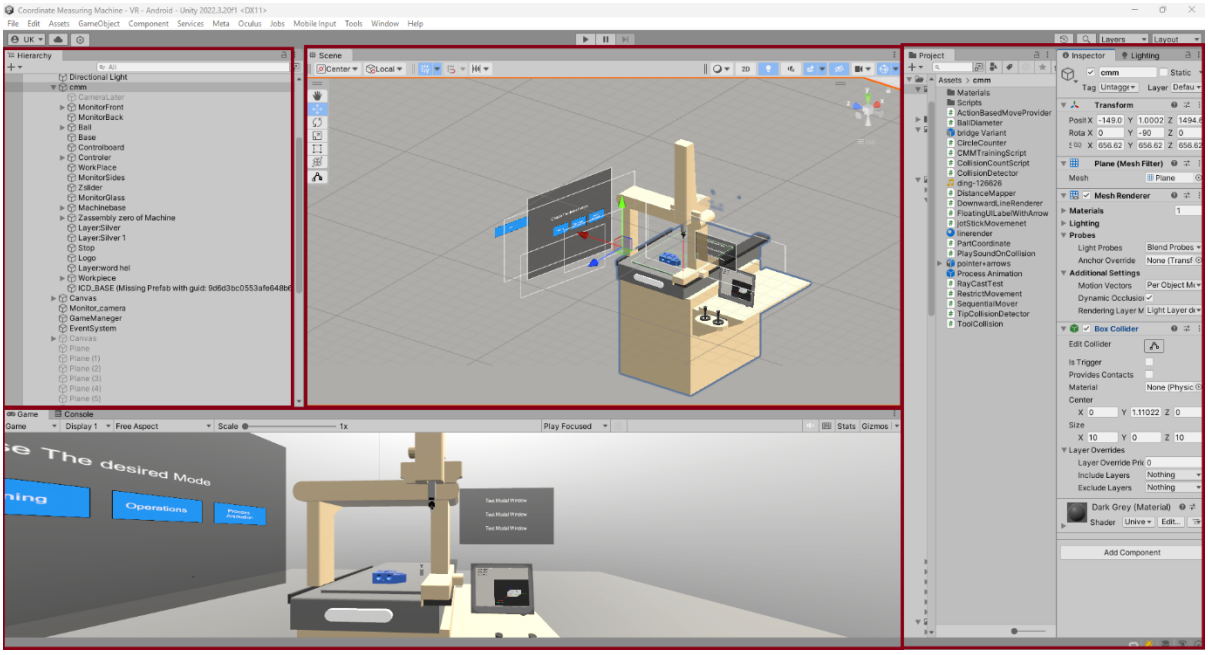


Figure 1.4: Virtual CMM interface developed in Unity

1.4 CMM Path Planning and Optimization Techniques

Path planning in coordinate measuring machines (CMMs) refers to determining an optimal sequence of measurement points and corresponding probe trajectories, a problem analogous to the Travelling Salesman Problem, where the goal is to visit all measurement features in the shortest

possible order. It plays a critical role in ensuring accurate inspection, collision avoidance, and efficient operation (Han et al., 2017). As manual planning is often time-consuming and suboptimal, there is a strong need for automated solutions, discussed in detail in Section 2.2.2.

1.5 Integration of VR Training and Path Planning

This thesis integrates a path planning module into the VR training environment, creating a unified platform where operators can learn CMM operation and interact with optimized inspection plans simultaneously. Figure 1.5 illustrates the complete pipeline: from CAD model input and feature extraction to algorithmic path optimization, and from visualization and validation in the VR environment to a ready-to-use program for the physical machine.

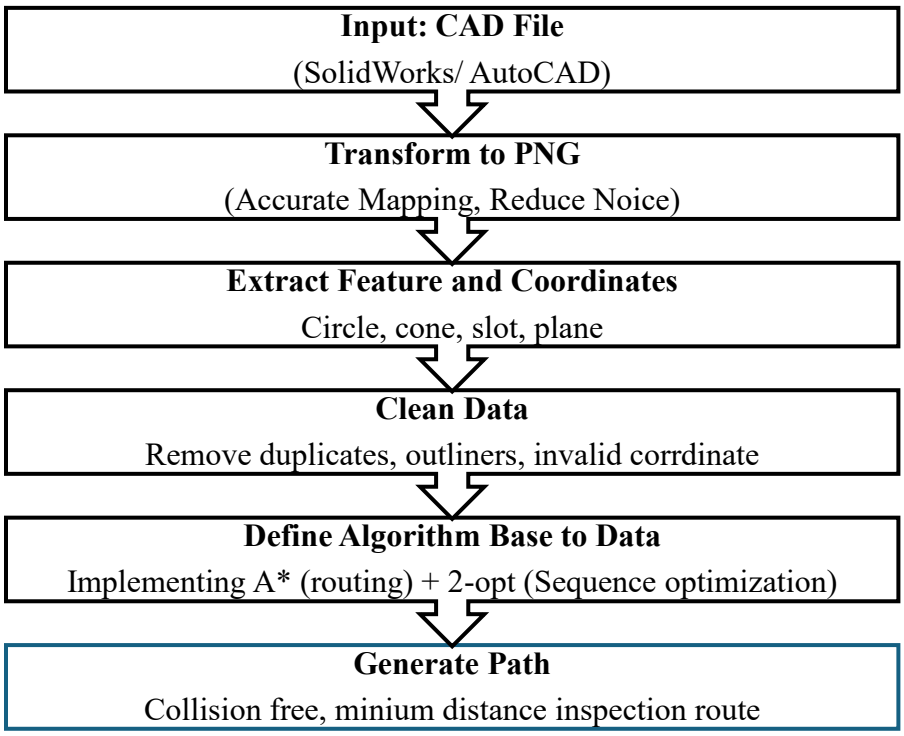


Figure 1.5: Path planning system

1.6 Research Motivation and Objectives

This research is motivated by the critical need to bridge the gap between theoretical metrology and practical skill acquisition in advanced manufacturing. CMMs are indispensable for verifying dimensional accuracy and geometric tolerances. Their high capital cost and mechanical delicacy

create significant barriers to training. These real-world constraints necessitate the development of high-fidelity, offline training solutions that allow for procedural mastery in a risk-free, sustainable environment. VR presents a transformative solution by providing an immersive "Virtual Twin" of the CMM. By leveraging the spatial awareness and engagement inherent in VR, trainees can develop technical competencies without consuming physical resources. The primary objective of this thesis is to develop and evaluate an immersive VR-based training system for CMM operations. The research addresses the following specific questions:

1. High-Fidelity Virtual Operation: How can the complex kinematic functions, calibration procedures, and user-interface interactions of a physical CMM be accurately replicated in VR to facilitate effective machine training?
2. Collision Avoidance: What computational methods are most effective for implementing real-time, collision-free path planning for the CMM probe within a dynamic virtual workspace?
3. Path Sequencing and Combinatorial Optimization: How can the measurement sequence be optimized to minimize travel distance? Specifically, can this be formulated as a Traveling Salesman Problem (TSP) and solved using heuristics, such as the 2-opt algorithm, to eliminate intersecting paths and improve efficiency?
4. Platform Efficacy in Engineering Simulation: What are the technical advantages and challenges of utilizing the Unity game engine for precision mechanical engineering applications? This involves assessing Unity's capacity to handle high-resolution CAD models and implementing custom algorithms (such as A* and 2-opt) to meet the reliability standards of an academic metrology lab.

These objectives are driven by three specific gaps identified in existing literature, examined in detail in Section 2.4: (1) the fragmentation of VR training platforms, many of them rely on outdated or proprietary toolkits; (2) the fidelity gap in training systems that skip metrologically critical operations such as PCS establishment and feature-fitting; and (3) the lack of integrated platforms where trainees can simultaneously learn CMM operation and understand path optimization.

1.7 Thesis Outline

This thesis is organized into the following six chapters.

Chapter 1 – Introduction outlines the research background, problems, objectives, and the thesis structure.

Chapter 2 – Literature Review examines existing work relevant to this research, including virtual reality applications in manufacturing metrology, current CMM path planning techniques, collision detection strategies, and measuring sequence methodologies. The chapter concludes with a summary that highlights the research gaps this thesis addresses.

Chapter 3 – Building a Virtual CMM System describes the development of the virtual CMM environment. It details the modeling of the CMM, CAD model importing and visualization, Unity-based functions for motion simulation, calibration procedures, collision detection mechanisms, and the teaching method implemented in the system.

Chapter 4 – CMM Path Planning focuses on the design and implementation of the proposed path planning approach. It explains the algorithm, optimization logic, and analytical methods used to evaluate the solution. Performance and accuracy considerations are also discussed.

Chapter 5 – Case Study presents the evaluation of the proposed approach using representative workpiece scenarios. Two cases are considered: a workpiece without obstacles and a workpiece with collision constraints. The results are compared with manual programming and existing methods to assess improvements in path efficiency, collision avoidance, and overall performance.

Chapter 6 – Conclusion and Future Work summarize the key findings and contributions, the research objectives achieved, and potential directions for future work.

Chapter 2. Literature Review

2.1 Virtual Reality in CMM Training

2.1.1 Virtual Reality

Virtual Reality (VR) entered engineering and manufacturing primarily as a simulation and visualization technology, originally motivated by the need to (i) understand complex spatial systems, (ii) reduce trial-and-error on expensive equipment, and (iii) enable safe rehearsal of risky operations (Küng et al., 2014). Early industrial "VR" often resembled what is now termed desktop simulation: the user interacted with 3D CAD models through a monitor using a computer mouse/keyboard and basic 3D navigation (Costello et al., 2021; Sadaoui et al., 2024). In manufacturing contexts, these desktop simulations supported offline programming, accessibility checking, and collision checking, and thus reduced machine downtime and risk (Eswaran & Bahubalendruni, 2022).

As computational performance and interaction devices matured, VR training evolved along two technical axes: (1) increased immersion and interaction fidelity (e.g., tracked controllers, stereoscopic headsets, and haptic devices) and (2) improved software integration, where multiple subsystems of geometry rendering, physics, control logic and procedural training could be combined in a single interactive environment(Eswaran & Bahubalendruni, 2022). By the 2000s, researchers began explicitly positioning VR as an educational tool in manufacturing, emphasizing its ability to enable trainees to interact with simulated equipment in ways not possible with static textbooks or videos. Zhao's VR-based CMM work, for example, emphasizes that traditional training documents are "plain and non-intuitive," and argues that VR enables users to manipulate virtual models and experience simulation outcomes, improving conceptual understanding and training flexibility (Zhao et al., 2022).

In parallel, formal education frameworks for coordinate metrology emerged in Europe most notably EUKOM (Europäisches Kompetenzzentrum für Koordinatenmesstechnik which translates as the European Competence Centre for Coordinate Metrology) and AUKOM (Ausbildung Koordinatenmesstechnik, i.e., a standardized training and certification program for coordinate metrology). These initiatives highlighted that competence in coordinate metrology requires

multidisciplinary foundations (mathematics, physics, manufacturing, design) and cannot be reduced to machine-specific pushing of buttons (Mian & Al-Ahmari, 2014). Weckenmann and Beetz (2006) positioned EUKOM as a response to the lack of standardized vocational training in coordinate metrology, advocating harmonized training content delivered through blended learning and internet-enabled collaboration.

More recently, VR training has become increasingly systematized and research-driven. A major shift is moving from isolated "virtual demonstrations" to structured learning environments informed by educational design. A large systematic review of immersive VR applications in higher education highlights that VR research has increasingly moved toward identifying design elements and evaluation methods, reflecting a maturation from "novelty demonstrations" to "learning effectiveness and design principles." This shift is directly relevant to modern machine training, including metrology: the emphasis is no longer merely on visual realism, but on whether VR systems deliver measurable learning outcomes and transferable skills (Radianti et al., 2020). In practical terms, VR machine training has been deployed through several platform categories:

1. CAD-integrated simulation platforms (e.g., Delmia V5 INSPECT, PTC Creo-based virtual measuring systems). These are typically strong in CAD interoperability, process planning, and representation of industrial workflows (Marjanovic et al., 2023; Mian & Al-Ahmari, 2014).
2. Custom VR development toolkits and virtual environment engines, including proprietary tools such as Eon Studio used in early virtual CMM systems (Chen et al., 2004).
3. Web-based 3D/VR platforms, which relied on technologies such as Java3D and browser applets to provide remote training and visualization(Lin et al., 2001).
4. Game-engine-based XR platforms, particularly Unity and Unreal Engine, which provide a unified environment for real-time rendering, physics, user interaction, and deployment to many devices (Jerald, 2015).

A modern technical enabler for cross-platform VR is the standardization of XR interfaces. OpenXR, developed by Khronos, was designed to provide a common API for developing XR applications across multiple devices, reducing the need for device-specific implementations and improving portability (The Khronos Group, 2017). This matters for training systems because education and industrial users often operate heterogeneous hardware ecosystems. A training

system that depends on a specific headset vendor, driver stack, or deprecated runtime becomes difficult to maintain and scale. Therefore, cross-platform programming and standards-based deployment are increasingly part of the "engineering requirement" for VR training systems (Geng et al., 2022).

Similarly, Neamţu et al. (2012) emphasized that modern measurement equipment, such as CMMs, imposes high competence requirements and that effective training must combine theoretical knowledge with practical skill development. Their review positions simulation as a valuable complement to hands-on practice, arguing that simulation-based training enables experimentation in a risk-free environment and allows repeated practice before using the real machine, thereby protecting expensive equipment and improving learning efficiency. This view is further supported by more recent work (Brunzini et al., 2023).

A core limitation of traditional training is operator-driven variability. A foundational insight in coordinate metrology education is that measurement outcomes are strongly influenced by human decisions. EUKOM's analysis of training needs emphasizes that many errors and uncertainties are not purely machine-driven; they arise from operator choices, the environment, and workpiece handling(Marxer et al., 2004). This implies that training must address more than "machine button knowledge." It must teach the reasoning behind probe selection, coordinate alignment, and feature construction.

This context explains why simulation and VR have become attractive, as they can teach both procedural steps and decision-making logic without incurring the risks and cost of repeated trials on real machines (Knoke & Thoben, 2021).

2.1.2 VR Training Performance

VR technology offers a transformative solution by immersing trainees in a simulated environment that closely mimics real operation. A well-designed CMM VR training system can eliminate the risk of physical damage. Trainees can freely experiment with machine controls in the virtual world without risking breaking a probe or crashing axes. Mistakes become learning opportunities instead of costly accidents. For example, a virtual CMM can enforce safe clearances if a trainee drives the

probe too close to a part, a collision may be shown in simulation, but no real hardware is harmed. This allows novices to build confidence and expertise in a zero-risk setting.

Because training is done offline, the real CMM remains available for production. Companies reduce downtime and avoid tying up expensive equipment for education (Bordegoni et al., 2022). Apart from an initial investment in VR software/hardware, the marginal cost of training additional operators is very low. VR is also highly scalable, multiple trainees can practice simultaneously on different virtual stations, even remotely (Bernal et al., 2022; Bunduwongse & Tangjitsitcharoen, 2025). This flexibility extends to scheduling and location: trainees can learn at their own pace, on their own schedule, without being limited by an instructor's or a machine's availability.

Modern VR provides users with immersive 3D experience and interactive feedback, which is far more engaging than reading manuals. In a virtual CMM, the trainee can see a lifelike 3D model of the machine and workpieces, operate a simulated joystick or controller, and observe probe movements and measurements in real time. Such systems can simulate realistic machine behavior including kinematics, sensor feedback, and even common errors. Research indicates that these attributes lead to better training outcomes. Trainees who learn in an interactive simulation demonstrate higher skill retention and make fewer errors when moving to real equipment (K. Zhou et al., 2021). In fact, comparative studies have found that VR training can be “more effective than traditional methods” by providing precise simulations and even enabling collaborative training scenarios not possible in a classroom (Asad et al., 2023). Overall, VR-based training delivers a safe, cost-effective, and pedagogically rich environment that addresses the major shortcomings of conventional CMM training.

2.1.3 Early Developments and Pioneering Work

1. **Human-in-the-loop path “teaching” in 3D space:** The inspection path is generated by pointing a virtual probe at a CAD model using a haptic device, mimicking “teach pendant” programming(Wang et al., 2009).

2. **Collision detection optimized for interactive use:** Haptic Virtual CMM (HVCMM) uses surface voxel representations for quick collision detection and adds force feedback based on a mechanics model when contact occurs (Chen et al., 2004).

Early systems were powerful but specialized. Haptic devices add realism but increase cost and integration complexity. Furthermore, haptic path planning emphasizes low-level probing and collision avoidance but does not provide comprehensive educational scaffolding (structured training modules, UI guidance, assessment tracking) unless explicitly added (Knoke & Thoben, 2021). While HVCMM demonstrated the potential of immersive interaction, EUKOM (mid-2000s) demonstrated the institutional need for harmonized training and web-enabled learning structures. EUKOM's emphasis on internet-supported delivery anticipated modern distributed learning environments, where trainees may not have local access to CMM labs (Marxer et al., 2004). This period is important because it clarifies what "training" must include: not just motion practice, but also measurement strategy, tolerance reasoning, and interpretation across competence levels.

The concept of virtual CMM (VCMM) has been explored in research for over two decades. One early example is the work of Hu and Yang (2009), who developed a VCMM platform to simulate CMM's operations and measurement processes in a virtual environment (Gąska et al., 2019; Küng et al., 2014). Their system enabled offline CMM programming and even performed error analysis and uncertainty evaluation virtually, effectively acting as a digital twin of the physical machine(Gąska et al., 2017). Around the same time, other researchers focused on specific aspects of virtual CMM design. Zhou et al.(2022) built a 5-degree-of-freedom parallel-link CMM in a virtual environment using OpenGL graphics and implemented an Axis-Aligned Bounding Box (AABB) algorithm for collision detection. This enabled the simulation to detect probe collisions with the workpiece or the machine structure in real time, an essential feature for safe training. Another influential effort was by Yang and Chen (2004), who introduced a Haptic Virtual CMM (HVCMM) for training. In their system, a force-feedback haptic device (essentially a joystick with tactile feedback) let the user "feel" interactions as if holding the CMM's probe. This innovation allowed trainees to practice guiding the probe along surfaces and features using tactile feedback, greatly enhancing realism. Chen and colleagues(2004) were among the pioneers in haptic CMM simulation, demonstrating that integrating tactile feedback helps users learn delicate manual probing motions with less trial-and-error (Ma et al., 2002). A second milestone is the VR-based CMM training system developed by Zhao and Peng (L. Zhao & Peng, 2011) and expanded in Zhao's thesis (2012). Zhao's work explicitly positions VR as a solution to costly, time-consuming

operator training, arguing that VR provides an intuitive environment where trainees can practice CMM operation before operating the real machine, while reducing collision risk and improving flexibility. The thesis states that the virtual CMM system enables users to perform basic CMM operational functions, reduces unnecessary collisions, protects sensors, and improves machine utilization by enabling offline verification. Importantly, it concludes that improving inspection efficiency ultimately requires integrating path planning theory into the virtual CMM system, an observation that directly motivates the gap addressed in the present thesis. Zhao's implementation relied on Eon Studio and VB scripting, which was feasible at the time but required specialized toolchains, reduced portability, and could become difficult to extend or maintain as industry VR ecosystems evolve. This becomes a key comparative point when motivating Unity-based solutions: modern engines provide integrated physics, UI, XR support, and broad deployment options, while older platforms often fragment these capabilities.

Neamtu et al. (2012) reviewed the use of Delmia V5 INSPECT for training in coordinate metrology and explicitly noted that while many measuring software packages support offline programming, only a smaller number simulate the complete measuring environment, including the CMM, human operator, and robotic loading/unloading systems. Their framing is important because it broadens the training objective: it is not only about probe movement but also about understanding the inspection workflow in a manufacturing system, including part handling, operator positioning, and process integration. Such solutions are powerful but often expensive, license-intensive, and tied to proprietary industrial CAD ecosystems. They also tend to prioritize process simulation and offline programming over immersive VR interaction and game-like instructional design.

In the late 2010s, a web-based virtual CMM learning environment was proposed by using browser-compatible technologies for inspection (Sang et al., 2018; Wang et al., 2011). It describes an interactive platform based on Java3D and Java Applets, allowing students to view the CMM structure from any angle and practice measurement procedures remotely while supporting many students online simultaneously. Web3D approaches greatly improve accessibility but have historically relied on technologies (e.g., applets) and typically offer limited physics realism and device-interaction fidelity compared to modern XR engines.

By the early 2020s, the literature shows two major evolutions:

1. **Digital Measuring Twins (DMT):** Marjanovic et al.(2023) described modeling and simulation of a virtual measuring system based on a real CMM using PTC Creo, generating DMIS programs, and validating them against real measurements. Their contribution emphasizes interoperability issues and highlights that simulation can support collision-avoidance prediction and the verification of measurement processes.

2. **Intelligent measurement and automation:** Cheng et al. (2023)proposed combining vision sensors with trigger probes and deep learning (YOLO) for feature detection, ant colony optimization for measurement sequence, and automatic measurement path planning to reduce manual intervention.

These works demonstrate that CMM research is increasingly focused on automation and intelligence, rather than just training. However, they also reinforce the gap: intelligent algorithms are often implemented in research prototypes or production-optimization contexts, rather than integrated into VR-based training platforms where operators can learn to understand and validate algorithmic plans visually and interactively.

Beyond fully virtual setups, some researchers have explored augmented reality (AR) for CMM training. AR systems overlay virtual guidance or annotations onto the CMM's real-world view. For instance, an AR-based training module might use a tablet or AR glasses to project step-by-step instructions or highlight measurement features on the actual machine during operation (Aromaa et al., 2019). Early AR prototypes in metrology have shown promise in reducing learning time by providing intuitive, on-site guidance, though they are less common than pure VR approaches in the literature (Taylor et al. 2019; Lin et al. 2023). Together, these pioneering works established the feasibility of using interactive simulations to teach CMM operations using visual, physical, and even haptic methods without a live machine.

2.1.4 Architecture of Modern VR Training Systems

Over years of development, a consensus has emerged that an effective VR-based training platform must model four key facets: geometry, physics, behavior, and rules (Bordegoni et al., 2022;

Brunzini et al., 2023). Geometry refers to the 3D models of the CMM and its environment, the machine's structure, probe stylus, workpieces, and any fixtures, created with CAD tools at a realistic scale and detail. Physics means simulating the physical laws and constraints: the kinematic motion of CMM axes, probe dynamics, gravity, and collisions. For example, the virtual probe should obey the machine's real motion range and speed limits and trigger a collision event if it intersects a workpiece model (Zhao et al. 2022). Behavior encompasses the dynamic responses and interactive feedback of the system, essentially the "live" aspects of the simulation. This includes the control logic for the CMM motors to respond to user inputs, sensor feedback (point measurements, probe hits), and any automated behaviors (e.g. moving the probe to a commanded coordinate). The behavior model ensures the virtual CMM acts and responds like a real one, reinforcing correct operational habits (such as homing the machine, calibrating the probe, etc.). Finally, rules represent the procedural and safety constraints of operation. The simulation enforces measurement protocols and sequencing rules so that trainees learn proper procedures. For instance, the system may require the trainee to perform a probe qualification (calibration) before taking measurements, or it may prohibit moving the probe without first establishing a part coordinate system (PCS), mirroring the rules of real CMM usage (Xu et al. (2022); Zhang et al. (2025)). Most training systems can render geometry, but metrology training requires geometric reasoning, feature construction from probe hits, datum enforcement, and the application of Geometric Dimensioning and Tolerance (GD&T). Feature construction uses mathematical formulations such as:

Plane fitting (three-point definition or least squares plane fit):

$$n = (p_2 - p_1) \times (p_3 - p_1), \quad n \cdot (x - p_1) = 0 \qquad (2.1)$$

Line fitting (two-point direction or regression):

$$d = p_2 - p_1, \quad x(t) = p_1 + t\,d \qquad (2.2)$$

Circle construction (three-point circumcircle):

A circle passing through three non-collinear points (x_1, y_1), (x_2, y_2), and (x_3, y_3)can be defined using the general circle equation:

$$x^2 + y^2 + Dx + Ey + F = 0 \qquad (2.3)$$

The coefficients D, E, and Fare obtained by substituting the three points into the equation and solving the resulting system. The circle's center $(\mathrm{h}, \mathrm{k})$and radius r are then given by:

$$\mathrm{h} = -\frac{\mathrm{D}}{2}, \mathrm{k} = -\frac{\mathrm{E}}{2}, \ \mathrm{r} = \sqrt{\left(\frac{\mathrm{D}}{2}\right)^2 + \left(\frac{\mathrm{E}}{2}\right)^2 - \mathrm{F}} \qquad (2.4)$$

This approach reconstructs a unique circle from three measured points, provided they are not collinear.

A key gap in many VR training tools is that they emphasize probe motion without fully implementing the mathematical feature-fitting and tolerance-evaluation pipeline required for operator competence. For example, a bridge CMM has the following limitations:

- axis travel limits,
- motion scaling,
- reference homing behavior, and
- coordinate mapping between machine and part systems.

The machine-to-part coordinate transformation is typically as follows

$$p_{PCS} = \boldsymbol{R}^T(p_{MCS} - \boldsymbol{o}) \qquad (2.5)$$

where $\boldsymbol{o}$ is the part origin in machine coordinates, and $\boldsymbol{R}$ encodes the PCS frame's orientation relative to the Machine Coordinate System (MCS).

A persistent gap in prior VR CMM systems is that many did not expose this transformation explicitly to the trainee. Instead, they either abstracted the establishment of the coordinate system or treated it implicitly. Yet operator errors in alignment are among the most common real-world metrology mistakes, making PCS establishment a core training requirement(Trapet & Waldele, 1994).

Table 2.1: Traditional vs. VR-Based CMM Training

Aspect	**Traditional Training (Classroom/Hands-on)**	**VR-Based Training**

Safety Risk	High - Novice mistakes can crash the probe or damage the machine. Safety protocols limit hands-on practice.	Virtually- Zero mistakes result in no physical damage. Collisions are simulated, allowing risk-free learning.
Cost & Resource Use	Significant - Requires dedicated CMM time, instructor supervision, and possible equipment repairs. Training interrupts production.	Lower incremental cost - Once developed, multiple trainees can use the VR system without taking a real machine offline. Reduces machine downtime and maintenance costs.
Scalability	Limited - One trainee (or a small group) per machine/instructor. Difficult to schedule; constrained by location and equipment.	High - Many trainees can train in parallel on separate VR setups. Flexible scheduling and remote training are possible.
Interactivity & Realism	Moderate - Relies on manuals, videos, and occasional supervised machine use. Learning is abstract until applied to the machine.	High - Immersive 3D interaction with a realistic CMM model. Trainees actively perform operations (homing, probing, etc.) with real-time feedback.
Learning Effectiveness	Variable - Steep learning curve; errors on real equipment can be discouraging. Knowledge retention depends on limited hands-on time.	Improved - Engaging, game-like environment increases motivation. Studies show VR trainees achieve skills faster and with fewer errors on real CMMs.

To implement such a VR-integrated simulation, various software tools have been utilized over time. Early systems often combined multiple specialized packages, for example, CAD software for geometry, a physics engine for collision detection, and custom code for control logic, which made development complex (Anagnostakis et al., 2017). In recent years, game engines like Unity have emerged as a powerful one-stop solution for VR simulations. Unity provides a flexible platform that can handle high-quality 3D rendering (geometry), realistic physics simulation, scripting of interactive behaviors, and rule-based logic, all within a single framework (Bernal et

al., 2022). Researchers have highlighted Unity's versatility for CMM training: it supports advanced physics (via built-in PhysX engine) to model probe contacts, allows custom C# scripting to define machine kinematics and measurement processes, and can enforce logical sequences of operations through its programming environment(Neamţu et al., 2012). Additionally, Unity's support for VR devices (headsets, trackers) and cross-platform deployment make it convenient to develop immersive training modules that run on various hardware (PCs, VR headsets, even mobile devices). Table 2.1 summarizes the key differences between traditional training approaches and modern VR-based training solutions for CMM operations.

2.1.5 Integration of VR Training and Optimization

Despite the advancements in virtual CMM training, there remains a notable gap in fully integrating realistic training with automation. Most VR-based CMM simulators to date focus on replicating manual operations, teaching the user how to control the machine and measure parts correctly. On a separate track, research on CMM automation (discussed in Section 2.2) has developed algorithms to optimize probe paths and measurement sequences, thereby improving efficiency. However, very few systems integrate these two aspects into a single unified simulation tool. In other words, existing VR CMM training systems lack advanced AI-driven planning modules that could teach not only operational procedures but also optimal measurement strategies. Likewise, many algorithmic path planning studies remain theoretical or tested in isolation, without being deployed in a trainee-facing simulation environment. A truly comprehensive solution would merge the immersive VR training platform with an automated path planning engine "under the hood," so that trainees can learn both how to operate the CMM and how to optimize its use. Trainees could practice generating inspection plans, receive guidance or automated suggestions for efficient measurement sequences, and immediately see the outcomes in the virtual environment. Such integration would address the disparate nature of current tools by encapsulating all facets of CMM operation and planning in a single simulation (Marjanovic et al., 2023). Unity's environment is well-suited to this, as it can incorporate optimization algorithms (via scripting or plugin libraries) alongside the VR interface. The following Section reviews the state of the art in CMM path planning and optimization strategies, highlighting those methods that complement VR training, and set the stage for this research to build an integrated VR training system and path planning optimization.

2.2 Path Planning for CMM Inspection

CMM inspection efficiency and accuracy heavily depend on the probe's measurement sequence and path, where poor planning increases time, errors, and machine wear while requiring expert effort. Automating path planning optimizes inspection time, improves accuracy and safety, reduces programming effort, and ensures consistent, efficient measurement across complex parts.

2.2.1 Problem Definition – TSP Analogies and Collision Constraints

In abstract terms, the CMM path planning problem can be viewed as a combination of a combinatorial routing problem and a motion planning problem. The first sub-problem is often likened to the Travelling Salesman Problem (TSP) (Zakharov et al., 2017). Each required measurement point on the part can be thought of as a "city" that the probe (the "salesman") must visit exactly once. In other words, if one must visit 10 cities and wants the shortest possible road trip, the CMM probe faces the same problem, but with measurement features instead of cities. The goal for sequencing is to find an order of visiting all points that minimizes the total travel distance (or time). This sequencing problem is NP-hard, meaning that as the number of features grows, the time needed to find the perfect answer grows so fast that it becomes practically unsolvable by brute force; and the number of possible sequences grows factorially with the number of points, making exhaustive search impractical for anything but small feature sets(Teodor et al., 2019). The second sub-problem is to ensure that the trajectory between consecutive pairs of points in the chosen sequence is collision-free. The CMM's probe and stylus must move through 3D space without hitting the part or any surrounding objects. Unlike a simple TSP on a plane, the shortest path between two points may not be a straight line if an obstacle is in the way that the path planner might need to insert detours or clearance moves (sometimes called via points or collision avoidance points) to route around objects (Z. C. Lin & Liu, 1997). In addition, a CMM has physical constraints (like axis travel limits and probe rotation angles) that define its configuration space. Thus, CMM path planning is a constrained optimization problem: it seeks the shortest route that visits all measurement targets while obeying kinematic constraints and avoiding collisions in a 3D environment (J. Zhang et al., 2003). The task can be divided into two parts: (1) Measuring Sequence Optimization, i.e., finding the optimal order of visits (solving a TSP-like problem on the set of feature points), and (2) Trajectory Planning for each move (finding a short, collision-free

path between consecutive points). Both parts are interrelated, an order that is optimal in pure distance may not be optimal once collision avoidance is considered, and vice versa. The presence of obstacles means that simply computing a shortest path through the points in free space (as TSP assumes) is insufficient; one must also consider the extra travel required to go around objects or to reorient the probe. Additionally, collision detection is a major requirement for both planning and training. Chen et al.(2004) emphasized that collision checking and collision-free probe path generation are central to offline programming and noted that prior VCMM tools using mouse/keyboard were inconvenient and slow for complex parts, motivating haptic interaction and efficient collision-detection approaches such as surface voxel representations.

The collision detection in CMM literature typically includes the following methods:

- coarse bounding volumes (e.g., AABB) for fast broad-phase detection,
- voxel or occupancy models for fast contact evaluation,
- ray tracing / swept volume checks for accurate clearance analysis.

Most CMM users treat collisions as simple "touch trigger," lacking an approach strategy (e.g., safe clearance planes, retract moves, probe orientation constraints). It is necessary to provide collision prediction and near-miss detection in addition to collision notification. This requires CMM path planning for combinatorial optimization and motion planning(Han et al., 2017).

2.2.2 Path Planning Methods

Broadly, methods in literature can be grouped into: (i) Rule-based and heuristic approaches, (ii) Graph-based and AI search algorithms, (iii) Metaheuristic optimization algorithms, and (iv) Hybrid and advanced methods that combine elements of the above or incorporate machine learning. The following are the details of each of these methods:

1. Rule-Based and Heuristic Approaches: Early industrial solutions and research prototypes often relied on rule-of-thumb strategies or simple heuristics derived from expert knowledge. For example, a commonly used greedy heuristic is the Nearest Neighbor (NN) strategy: start at an initial point (often the first feature or an arbitrary origin), then go next to the closest not-yet-measured point(Y. J. Lin et al., 2001). Lin and Chen (1997) applied a nearest-neighbor method

followed by a refinement pass to improve the measuring sequence. The NN heuristics are fast to execute and easy to understand, but they can get trapped in suboptimal tours. Another rule-based approach is to group measurements by locality or by feature type (e.g., measure all features on one plane before moving to the next), reflecting how an expert might manually plan to minimize travel or probe changes. Knowledge-based systems have also been proposed, in which the rules for sequencing and orientation are encoded from expert domain knowledge. For instance, some systems interface with CMM programming software (like PC-DMIS or Siemens NX CMM module) and provide recommendations based on if-then rules (e.g., "if two holes are close and share an orientation, measure them consecutively"). These rule-based approaches are intuitive and can yield decent results in simple cases, but they do not guarantee optimality. They might miss opportunities for improvement in complex layouts and are limited by the completeness of the encoded knowledge. As parts and inspection plans grow in complexity (with many features scattered in 3D), purely heuristic methods may produce sequences far from the true optimum (Teodor et al. 2019; Abdulhameed et al. 2020). Nonetheless, they form a baseline and are often very quick to compute, making them useful in practice for a first-pass solution or in time-sensitive scenarios.

2. Graph-Based and AI Search Algorithms: In these methods, the planning problem is represented as a graph or state-space search, and classical AI search algorithms (like A, depth-first search, or breadth-first search) are employed to find solutions. One approach is to construct a graph where nodes represent measurement points (or specific probe configurations at those points), and edge weights represent the cost (distance or time) of travelling between points along a collision-free path. The problem of finding the minimal tour that visits all nodes is then a graph search problem, which can be tackled with algorithms such as A (which uses heuristics to prune the search) or other optimal search techniques. For example, Li et al. (2016) modeled adjacent feature relationships in a graph and used a greedy search on a Voronoi diagram to plan a near-optimal path. Similarly, graph traversal algorithms have been applied to sub-problems, such as finding the shortest safe path between two points, a typical robotics problem(Han et al., 2018). Depth-first search (DFS) or breadth-first search (BFS) can systematically explore possible paths on a discretized representation of the CMM's workspace, though these uninformed searches are usually too slow without heuristics. A* (A-star) is more commonly used because

it guides the search using an estimate of remaining distance, significantly improving efficiency in finding shortest paths through a grid or network. Graph-based methods inherently handle collision avoidance by integrating obstacle information into the graph, for instance, sampling points around obstacles or using configuration-space nodes that encode specific probe orientations to avoid collision. The downside is that as the environment's complexity grows, the graph can become enormous, especially when accounting for probe rotations in different states, which makes pure graph search computationally expensive (Zhao et al. 2022). Moreover, the process of modeling the environment (obstacle geometry, etc.) into a graph can itself be complex and time-consuming (Hart et al., 1968). Despite these challenges, graph search techniques guarantee the existence of a path and can find the optimal one given enough time, which is a strong advantage for the trajectory-planning portion of the problem. Some researchers have combined graph searches for local point-to-point moves with higher-level heuristics for sequencing. For example, one might use A to find the shortest collision-free path between any two feature points and then feed those distances into a TSP solver to obtain the sequence, thereby separating collision avoidance (handled by A) from sequence optimization.

3. Metaheuristic Optimization Algorithms: Given the NP-hard (Non-deterministic Polynomial-time hard) nature of sequence optimization, a large body of work applies metaheuristic algorithms, stochastic search methods inspired by natural processes, to find near-optimal CMM paths. Genetic Algorithms (GA), which mimic biological evolution, were among the first such methods applied to CMM inspection planning. Qu et al. (1998) formulated path planning as an optimization problem and used GA to evolve better measuring sequences. Lu et al. (1999) reported that the GA-planned inspection routes reduced the total travel distance by about 26–33% compared to the original manual path in their tests, concluding that the GA approach was significantly more effective for complex inspection tasks. Simulated Annealing (SA), which simulates the cooling of metals to escape local minima, has also been used to refine CMM sequences (Sadaoui et al., 2024; Sang et al., 2018). Ant Colony Optimization (ACO), inspired by how ants find shortest paths, has demonstrated strong performance on path-finding problems and has been adapted for CMM planning. For instance, Han et al. (2017) proposed an improved ACO algorithm that aims to enhance CMM inspection efficiency. By adjusting the pheromone update rules, their ACO variant found shorter paths and avoided collisions more

effectively, demonstrating the potential of swarm intelligence in this domain. Particle Swarm Optimization (PSO), another swarm-based method, has been applied to related path inspection problems (e.g., scanning continuous curves) and could also be used for CMM point sequences. Some comparative studies on free-form surface inspection have shown that ACO can outperform GA and PSO under certain conditions, achieving the best balance between path length and speed(Z. C. Lin & Liu, 1997). Other methods appearing in CMM literature include Tabu Search, Bee Colony algorithms, and Hybrid Evolutionary algorithms, each with its own mechanisms to explore the vast search space of possible paths. The appeal of metaheuristics is that they can handle large problem sizes and complex objective functions, including multi-objective ones such as combining distance and probe rotations, without guaranteed optimality. They are typically easier to integrate with existing CMM software as well. One can take a list of points and apply a GA or ACO routine to reorder them, treating the path cost as a black box evaluation with collision penalties added to the cost if needed. The drawback is that these algorithms can be computationally intensive and may require careful parameter tuning (e.g., population size, number of iterations) to achieve good results in a reasonable time. However, with modern computing power, metaheuristic methods have become quite viable even for online (real-time) optimization in some cases (Teodor et al., 2019).

4. Hybrid and Advanced Methods: Recent research trends show a push towards hybrid strategies that combine the strengths of different methods, as well as the incorporation of machine learning for adaptive planning. One class of hybrid approaches addresses the two sub-problems (sequencing and local path) using distinct methods. For example, a promising approach uses a fast metaheuristic or heuristic to generate a good initial sequence, then uses a graph-based planner like A to compute the detailed trajectory for that sequence, including collision avoidance (Azevedo et al., 2024). If the trajectory for a given sequence is not collision-free, the planner can modify the sequence locally, e.g., using a 2-opt swap heuristic, a simple local optimization in TSP, or inserting avoidance moves as needed (Mian & Al-Ahmari, 2014). By iterating between sequence optimization and path adjustment, the algorithm can converge to a near-optimal collision-free tour. This concept of combining a routing algorithm with a motion planner (akin to A + 2-opt, for example) aims to produce solutions that are both efficient in terms of distance and feasible to execute. Another hybrid approach is hierarchical planning.

Yau and Meng (1995) proposed a hierarchical system where coarse movements on a complex surface were planned first, then refined locally, using a mix of heuristics and geometric algorithms.

Beyond traditional algorithms, researchers explore machine learning techniques for CMM path planning. One area is Reinforcement Learning (RL), where an AI agent learns to move the probe optimally through trial-and-error in a simulated environment. Some studies in related robotics domains have combined classical path planners with deep RL to handle dynamic environments or unexpected changes (Jaramillo-Martínez et al., 2024). For instance, methods have been proposed to replan CMM paths in real time when certain features need re-measurement in dynamic tasks, using Q-learning or guided policy search to decide the next move (Q. Zhou et al., 2024). However, applying deep learning to CMM planning is challenging due to the large number of training examples, which may not be readily available for diverse part geometries, and the difficulty of ensuring collision-free results without manual verification (Tamizi et al., 2023). Knowledge-based improvements remain relevant, too. For example, Zhang et al. (2000) employed a clustering algorithm on features to minimize probe orientation changes, thereby reducing one source of error, and then solved a shortest-path problem over those clusters(S. G. Zhang et al., 2000). Li et al. (2016) introduced an orientation-point relational schema to formally relate probe orientations to subsets of points and developed an algorithm that optimizes the number of probe reorientations and the path length. Their approach effectively treats probe orientations as another "dimension" of the planning problem, seeking a balance between fewer rotations and shorter distance. The results show improved efficiency, though their use of a greedy strategy across orientation groups could still yield a globally suboptimal final sequence. Some systems integrate human expertise via interactive or teaching-based planning. For example, Anagnostakis et al. (2017b, 2017a) developed a virtual CMM inspection planning tool that captures an expert's motion as they manually guide a probe virtually, then formalizes that strategy for reuse (Anagnostakis et al., 2017). This approach blurs the line between manual and automated planning, leveraging intuitive human planning for complex decisions while automating execution and recording.

Modern CMMs, especially those with articulating probe heads or fully robotic 5-axis CMMs, add another layer of complexity by requiring the selection of the probe's approach angle for each feature. A feature might be measurable from various orientations; choosing an optimal orientation

for measurement can reduce the need for extra moves or recalibrations. Frequent probe reorientations are known to degrade accuracy due to recalibration errors and kinematic uncertainties, and additional inspection time (Zhao et al. 2022). Researchers have thus treated probe orientation optimization as an important sub-problem. One strategy is to cluster measurement points by required orientation: for instance, to group all features that can be measured with the probe at angle A, measure them in one cluster, then rotate the probe to angle B and measure the next cluster, and so on (Abdulhameed et al. 2020). This minimizes the number of orientation changes. Zhang et al.'s clustering approach (2000)is an example that ensures the minimum number of probe rotations, although it doesn't simultaneously minimize path length within those clusters. Another approach (Li et al., 2016) is to define a relational schema that links each point to feasible probe orientations, then algorithmically select a set of orientations that cover all points with as few orientations as possible to optimize the path length. For five-axis CMMs that allow continuous 2-axis probe rotation while measuring, the notion of "orientation" becomes a continuous variable; planning algorithms must then account for the probe head's motion as part of the path. Some recent works have looked at path reuse and clustering in 5-axis scenarios. Zhao et al. (2022) tackled a realistic case of multi-station engine block inspection, proposing to classify measurement points based on their feasible orientation cones, the range of probe angles that can access the point, and cluster them to minimize distinct probe orientation. They then planned local paths for each cluster using a variant of the Rapidly-Exploring Random Tree (RRT) algorithm that accounted for path reuse, and finally solved the global clustering sequence using an enhanced GA. This comprehensive approach greatly reduced redundant probe motions: by reusing portions of paths for repeated features and adding new paths only when necessary, they achieved an over 50% reduction in total planning time for dynamic re-inspection tasks. It underscores how combining orientation optimization, clustering, and efficient local path planning can yield significant efficiency gains in complex 5-axis CMM operations (Morales et al., 2019).

This thesis emphasizes a hybrid strategy: global sequence optimization (TSP heuristic view) and local collision-free routing. Two algorithms are central:

- **A*** for shortest collision-free routing in discretized configuration spaces. A* is a heuristic shortest-path method for graphs, originally developed by Hart, Nilsson and Raphael; the

complexity and properties of admissible heuristic search (including A*) have been studied extensively in the literature (Xuan et al., 2025).

- **2-opt** for improving sequences as a local search method for TSP-style tours. The 2-opt swap removes route crossings by reversing segments and was proposed by Croes (1958).It remains one of the simplest and most effective local improvement heuristics.

Many published CMM planning methods either optimize the sequence or address collision avoidance, but do not integrate them into an interactive training platform where users can see the resulting path, validate it, and learn how planning affects measurement efficiency. This creates a gap between algorithmic research and operator training practice.

2.3 Comparison of Manual vs. Computer-aided Planning

The benefits of CMM path planning are often demonstrated by comparing it with manual programming. Manual plans, even from skilled users, tend to be conservative and can miss optimization opportunities that a computer algorithm can identify by exploring alternatives. Computer-aided methods can substantially shorten the travel distance and measure cycle time. In terms of programming effort, manual programming can take hours for a complex part, whereas a computer algorithm can produce a plan in minutes. Table 2.2 qualitatively compares typical outcomes of manual vs. automatic path planning.

Table 2.2: Manual Planning vs. Automated Path Planning

Outcome Metric	Manual Planning (Human Operator)	Automated Planning (Optimization Algorithm)
Programming Time	High - Manual sequencing is time-consuming, especially for many features. Each path and clearance move is defined by the programmer, often through trial and error.	Low - Once set up, algorithms compute sequences quickly from seconds to a few minutes for complex parts, greatly reducing offline programming time.
Inspection Cycle Time	Longer - Human-devised paths may not be globally optimal. Unnecessary travel and probe reorientations are common, extending the measurement time.	Shorter - Optimized sequences reduce total travel distance. Studies show a 20-30% reduction in path length/time compared with manual paths. Collisions are avoided through precise modelling, without excessive caution.
Path Optimality	Moderate - Depends on the users' skill. Good experts achieve decent results, but complex interactions (many points, obstacles) make it hard to manually find the true optimum.	Near-Optimal - Algorithms explore a vast solution space to find very efficient routes. Metaheuristics and hybrids can approach optimal solutions within a few percent of the theoretical minimum path length in many cases.
Collision Avoidance	Ensured by manual checks - Programmer inserts clearance moves and safe approaches based on experience. However, there is a risk of human oversight leading to potential collisions or near-misses.	Ensured by design - The planner explicitly checks for collisions (e.g., via collision-detection modules). Paths output by the algorithm are collision-free by construction and can even optimize the number of clearance moves needed.
Consistency	Variable - Different users may produce different paths for the same part, undermining consistency and affecting the standardization of cycle times and results.	High - Given the same input and criteria, the algorithm produces a consistent plan every time. This ensures standardized inspection processes and easier validation of results.
Adaptability	Low - Re-planning manually for design changes or new tolerance requirements takes significant additional effort.	High - Automated planners can quickly recompute paths if the part or feature list changes. Some advanced methods even support path reuse,

	Manual plans are not easily reusable beyond their specific scenario.	adapting segments of existing paths to new measurement tasks to save time.

2.4 Research Gaps

From this review of path planning strategies, a clear gap in the literature emerges: the need for faster, unified planning systems that can seamlessly integrate measurement sequencing and trajectory planning into a single, efficient solution. Many traditional approaches handle sequence optimization and collision avoidance in isolation or in sequence. For example, an algorithm might fix an order of points, then add detours to avoid collisions, which could be suboptimal, or, conversely, plan detailed paths between points assuming a pre-set sequence. This separation can lead to subpar results or slow iterative loops. Modern hybrid approaches are moving toward combining these steps, for instance, framing the problem as a Generalized Traveling Salesman Problem (GTSP) where each cluster of points (by orientation) must be visited, and each cluster has internal path options. The state-of-the-art study by Zhao et al. (2024) is a good example of integrating multiple aspects (orientations, path reuse, global sequence) with tailored algorithms. Still, such advanced solutions are relatively new and have mostly been demonstrated in academic case studies. There is still room to improve their generality applied to any part geometry, computational speed, and ease of use. Moreover, as noted earlier, a few of these sophisticated planners have been incorporated into training tools for human operators. In an industrial setting, an ideal scenario is a simulation-based training system that teaches new CMM programmers not only how to operate the machine but also how to leverage automated path planning effectively.

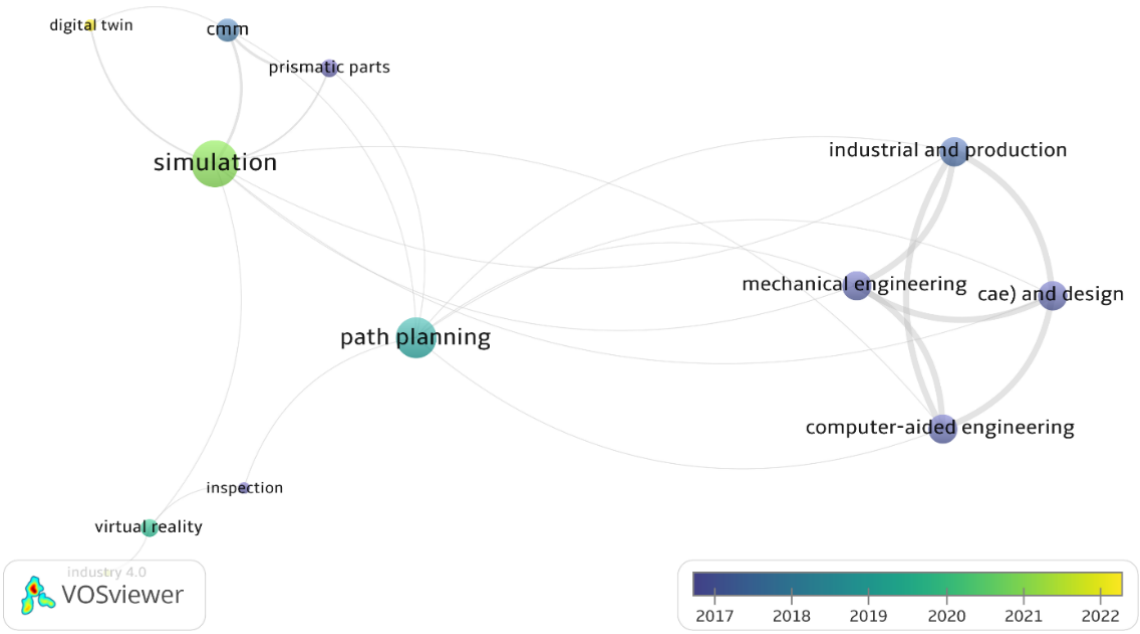


Figure 2.1: Keyword co-occurrence map of CMM-related research

Figure 2.1 presents a keyword co-occurrence map generated using VOSviewer from publications indexed in Scopus over the period 2017-2022. In this map, each node represents a keyword, node size reflects how frequently the keyword appears across the reviewed publications, and connecting lines indicate that two keywords co-occur within the same publication. Keywords that appear together frequently are positioned closer together and share stronger connecting lines. Three clusters are visible in the map. The first, positioned in the upper-left region, groups CMM, simulation, and digital twin, reflecting research focused on machine modeling and virtual replication. The second, in the centre, connects path planning and inspection, representing the algorithmic optimization literature. The third, on the right, clusters computer-aided engineering, mechanical engineering, and industrial and production engineering, reflecting the broader disciplinary context. Notably, 'virtual reality' and 'industry 4.0' appear as relatively isolated nodes with weak connections to the path planning cluster, despite being active research areas individually. This spatial separation in the map provides visual confirmation of the central gap this thesis addresses: VR-based CMM training and automated path planning are active but largely disconnected research streams. From this review, three specific gaps emerge in CMM operations and training systems:

1) **Fragmentation of platforms.**

Early VR CMM systems were built in specialized toolkits (e.g., Eon Studio + VB scripting) that are difficult to port and modernize. CAD-centric platforms (Delmia V5, PTC Creo) are powerful but expensive and may not deliver full immersive training with modern VR interactions. Web-based training improves accessibility but historically relied on deprecated runtimes and limited physics/interaction fidelity.

2) **Mechanical engineering fidelity gaps.**

Many training simulators emphasize visual representation and basic collision detection, but do not fully implement:

- Coordinate system establishment (PCS transformations),
- Probe qualification modeling,
- Feature fitting mathematics,
- Motion constraints and kinematic scaling in metrology units.

These gaps reduce skill transfer because real CMM competence depends on these metrology-specific behaviors.

3) **Limited integration with optimization and planning.**

Even VR training systems exist, they usually train manual operation; path optimization research is conducted in parallel but is rarely embedded in the training experience. Zhao's thesis(2011) explicitly identifies that path planning should be implemented in the virtual CMM system to improve inspection efficiency. Yet few published systems fully unify these components.

Table 2.3 consolidates these findings by comparing six representative prior works against the seven capability dimensions identified in this review. As shown, no single existing system satisfies all dimensions simultaneously; most achieve depth in one area (either training fidelity or path

optimization) at the expense of the other. This thesis is positioned to address the full set of dimensions within a single integrated platform.

Table 2.3: Research gap comparison matrix

Work / System	**Full VR / Immersive Interaction**	**Metrology Fidelity (PCS, feature fitting)**	**Path Planning Integration**	**Modern XR Platform (Unity / OpenXR)**	**Collision-Free Routing (A* / Graph)**	**Sequence Optimization (TSP / 2-opt)**	**Open / Cross-platform Deployment**
Chen et al. (2004) Haptic Virtual CMM	✓	~ Basic collision only	X	X Haptic device only	~ Voxel-based only	X	X Proprietary haptic SDK
Zhao & Peng (2011–2012) VR CMM Training (Eon Studio)	✓	~ Partial – no PCS training	X Identified as future need	X Eon Studio + VB scripting	X	X	X Deprecated toolchain
Neamtu et al. (2012) Delmia V5 INSPECT	X Desktop simulation only	✓ Full CAD-based metrology	~ Offline programming only	X Proprietary CAD platform	~	~	X License-intensive
Marjanovic et al. (2023) Digital Measuring Twin (Creo)	X CAD simulation, no VR	✓ DMIS program validation	~ Collision prediction only	X PTC Creo ecosystem	✓	X	X
Cheng et al. (2023) YOLO + ACO automated CMM	X	✓ Feature detection via YOLO	✓ ACO measurement sequence	X No VR / XR component	~	✓	X Research prototype
Han et al. (2017–2018)	X	~	✓ ACO	X	~ Pheromone	✓	X

Improved ACO for CMM			sequence optimisation		-based avoidance		
THIS THESIS — Khan & Peng VR CMM + A* / 2-opt (Unity)	✓	✓ PCS, stylus qual., feature fitting	✓ A* + 2-opt in VR environment	✓ Unity + OpenXR + XR Toolkit	✓ A* shortest path, clearance checks	✓ TSP heuristic + 2-opt refinement	✓ PC, VR headset, mobile

The path planning literature has seen strong progress, from early CAD-based automation and metaheuristic optimization to modern digital twins and deep learning-based measurement systems. However, key challenges remain:

- **Scalability and usability:** Advanced planning methods can be computationally heavy or require careful tuning (metaheuristics) and are often not packaged for intuitive operator validation.

- **Separation of "sequence optimization" from "trajectory feasibility":** Many methods treat routing and collision as separate steps, which may be suboptimal and complicate real-world use.

- **Validation and operator trust:** Modern AI-driven systems (deep learning feature detection + optimized sequences) can fail when assumptions break (e.g., occlusions) and require a robust validation environment. This motivates simulation-based verification integrated into planning and training.

This thesis's approach is positioned as a response to the above issues by proposing an integrated framework that:

- Uses Unity as an integration platform to unify geometry, physics, interaction, and rule enforcement (enabling full procedural training and extensibility), supported by modern XR interaction architecture (Interactors/Interactable/Interaction Manager).

- Leverages modern cross-platform XR standards such as OpenXR to enhance portability and maintainability across devices, reducing dependence on deprecated or proprietary VR ecosystems.

- Integrates path planning methods (A* routing and TSP/2-opt sequence refinement) into the same environment where trainees learn to operate the CMM, directly addressing the separation problem between training and planning research.

Therefore, this thesis aims to bridge that gap by developing a VR-based CMM training platform augmented with an intelligent path-planning module. The trainee can not only learn the manual skills required to operate a CMM but also interact with algorithms that suggest or generate optimized probe paths. Our approach builds on the ideas from literature, such as combining heuristic sequencing with graph-based collision checks, akin to a planner guiding a 2-opt improvement of a route. By integrating these into the Unity-based simulation, an interactive environment is built for experimentation with path optimization: for example, users can input a set of features, observe the algorithm's planned route, and even step through the execution virtually to understand how optimized plans differ from naive ones.

Chapter 3. VR-Based CMM Operation Training System Development

This chapter describes the development of an immersive Virtual Reality (VR)-based Coordinate Measuring Machine (CMM) operation training system using the Unity game engine. The central objective of the system is to provide a realistic, interactive environment where learners can practice core CMM operating procedures including stylus qualification (a calibration step where the probe tip is touched against a precision reference sphere to determine its exact size and position offset before any measurements begin), coordinate system establishment, and feature measurement, without requiring continuous access to physical metrology equipment. As established in Section 2.1.2, VR training directly addresses the access, cost, and risk barriers to conventional CMM training by providing a safe, scalable simulation environment.

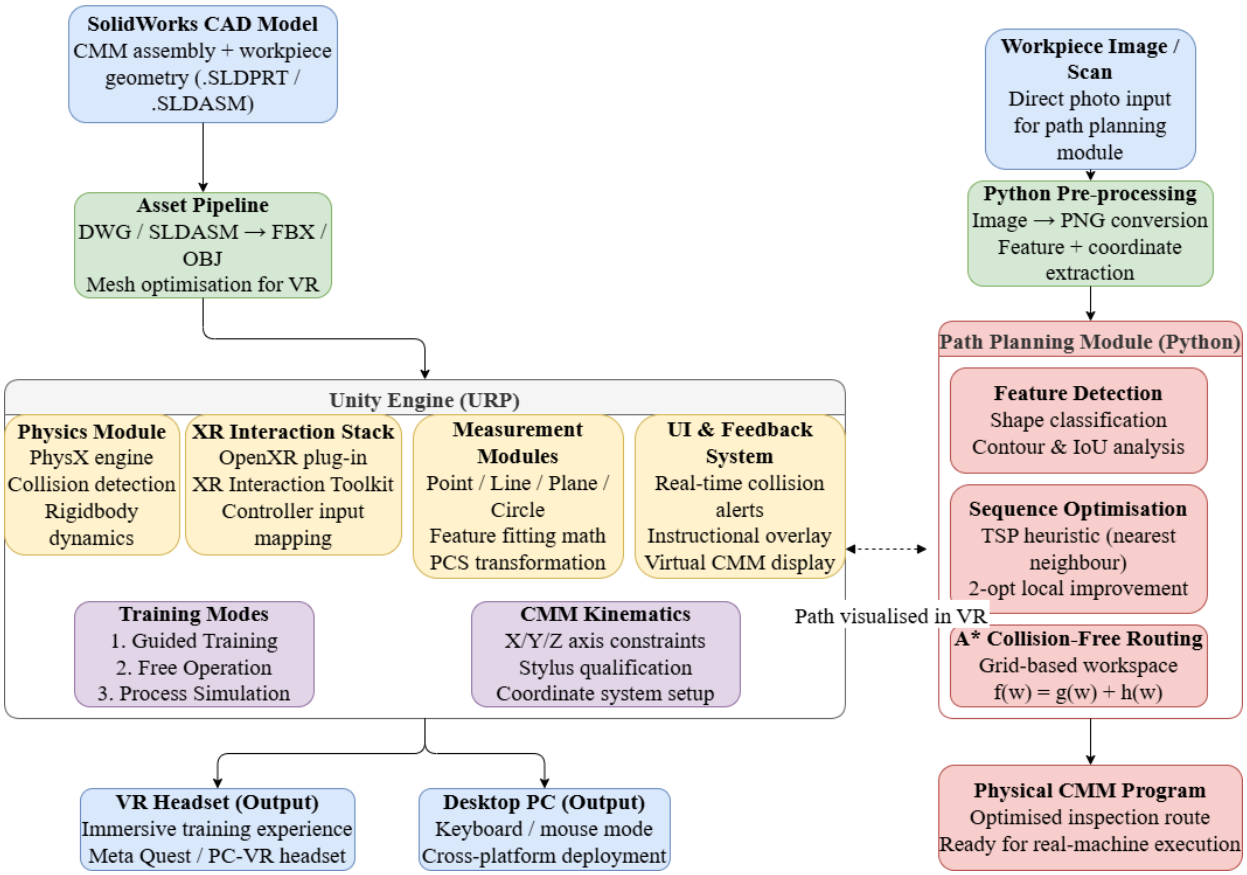


Figure 3.1: VR-Based Training System's Architecture Overview

Figure 3.1 presents the overall system architecture of the VR-based CMM training platform. The system consists of two integrated modules: a Unity-based VR training environment and a Python-based path-planning module. These components share data bidirectionally, allowing optimized probe paths generated offline to be visualized and validated within the immersive VR environment. The following sections describe each component in detail.

3.1 Objectives and requirements

The system requirements are defined for both pedagogical and technical needs:

1. **Pedagogical coverage of the CMM workflow.** The system must reflect the order and logic of real inspection workflows. The training process is structured for a standard CMM operational pipeline: powering the system, homing, stylus qualification, coordinate system selection, measurement execution, and reporting/analysis. This mirrors common instructional frameworks of coordinate metrology curricula and typical shop-floor routines.
2. **Interactive practice modes with increasing autonomy.** The application must support (i) guided training with on-screen instructional scaffolding, (ii) unguided operation for self-directed practice, and (iii) a process simulation mode where the system autonomously demonstrates complete operation sequences. This structure is implemented through a main menu with three modes, i.e., Training, Operations, and Process Simulation.
3. **High-fidelity geometric representation and credible machine behavior.** The virtual CMM must be visually recognizable (bridge, rails, spindle, probe, table, etc.) and behave like a Cartesian CMM: axis motion must follow constrained X–Y–Z kinematics; measurement points should be recorded only when probe contact occurs; and collisions should be detected robustly.
4. **Cross-platform and device adaptability.** The system can run on desktop PCs using keyboard/mouse for navigation, and VR headsets for immersive training. Unity is used to build applications for different platforms through its Build Settings workflow.
5. **Extensible measurement library.** The system includes implemented training for foundational measurement types (point, line, plane, circle). The system architecture, however, is extensible to support additional features commonly measured on CMMs (e.g., cones, slots, cylinders) by adding feature-fitting routines and measurement strategies to the same framework. Figure 3.2

illustrates the design requirements and system goals for the VR-based CMM training system, outlining the core functional specifications, performance criteria, and user-centred objectives that guide its development.

3.2 Development of toolchain and platform selection

3.2.1 CMM modeling and asset pipeline

The CMM geometry and training workpieces were modeled in SolidWorks to ensure dimensional realism consistent with mechanical engineering practice. The models were exported from CAD into mesh formats suitable for real-time simulation and rendering. Unity supports import of several widely used model exchange formats (including FBX and OBJ) as part of its standard asset pipeline. This capability enables a workflow in which CAD assemblies are tessellated into polygonal meshes and then imported as Unity GameObjects, preserving the CMM's major structural elements and relative dimensions.

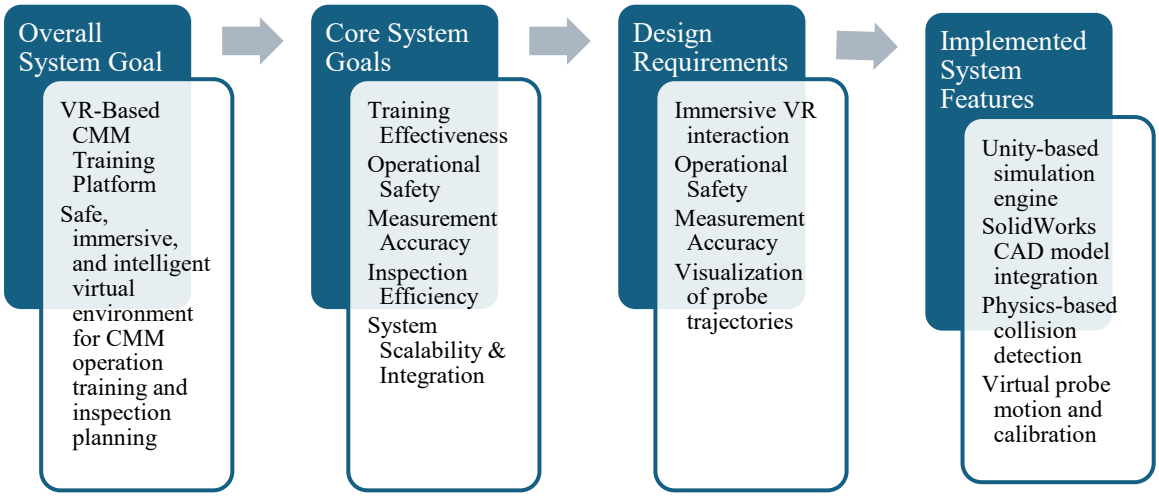


Figure 3.2: Design requirements and system goals

SolidWorks with its Open CASCADE Technology (OCCT) can model accurate parametric geometry. Real-time simulations require polygon meshes with manageable triangle counts. Therefore, the export process is guided by standard VR performance considerations: unnecessary fillets, micro-features, and internal geometry not required for training realism are reduced or omitted where appropriate, and colliders are generated with a level of fidelity that balances

collision accuracy and computational overhead. The VR-based CMM training platform is developed using the Universal Render Pipeline (URP - Unity's optimized graphics system for consistent visual quality across devices) in Unity to achieve a balance between visual realism and real-time performance. As shown in Figure 3.3, HDR rendering is enabled with 4× MSAA anti-aliasing to improve visual clarity, while LOD cross-fade with blue-noise dithering is applied to ensure smooth transitions between detail levels.

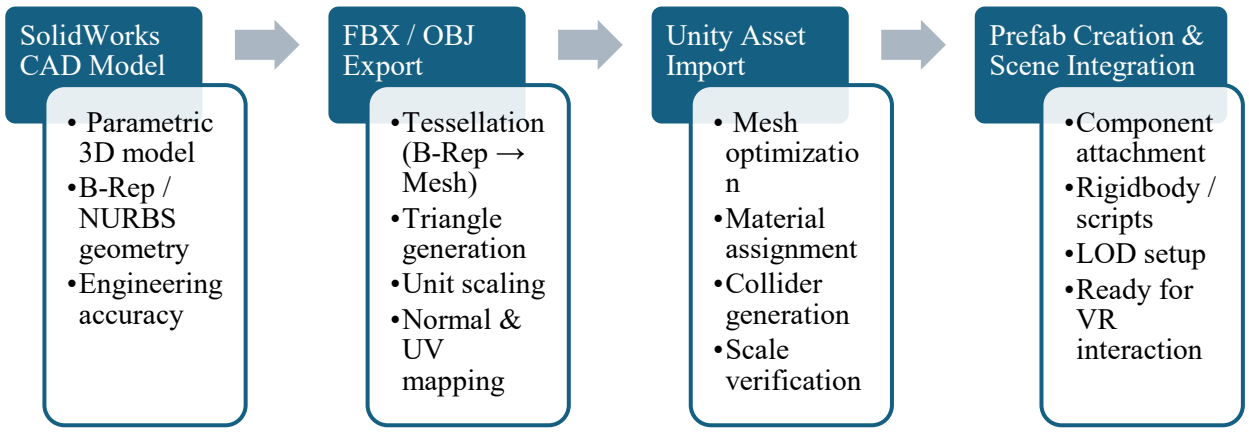


Figure 3.3: CAD-to-Unity asset pipeline

For lighting, the main and additional lights are configured in Per Pixel mode with shadow casting enabled, and high shadow resolutions (up to 4096) are used to enhance depth perception and spatial realism. A cascade shadow system with four cascades and a maximum shadow distance of 150 mm is applied to maintain shadow quality across varying viewing distances. Reflection probe blending and box projection are also enabled to improve material realism and environmental reflections. These settings, illustrated in Figure 3.4, are selected to provide the high visual fidelity required for accurate geometric perception and inspection tasks in the VR-based CMM training environment while maintaining stable real-time performance. Figure 3.5 shows the rendered model of the CMM.

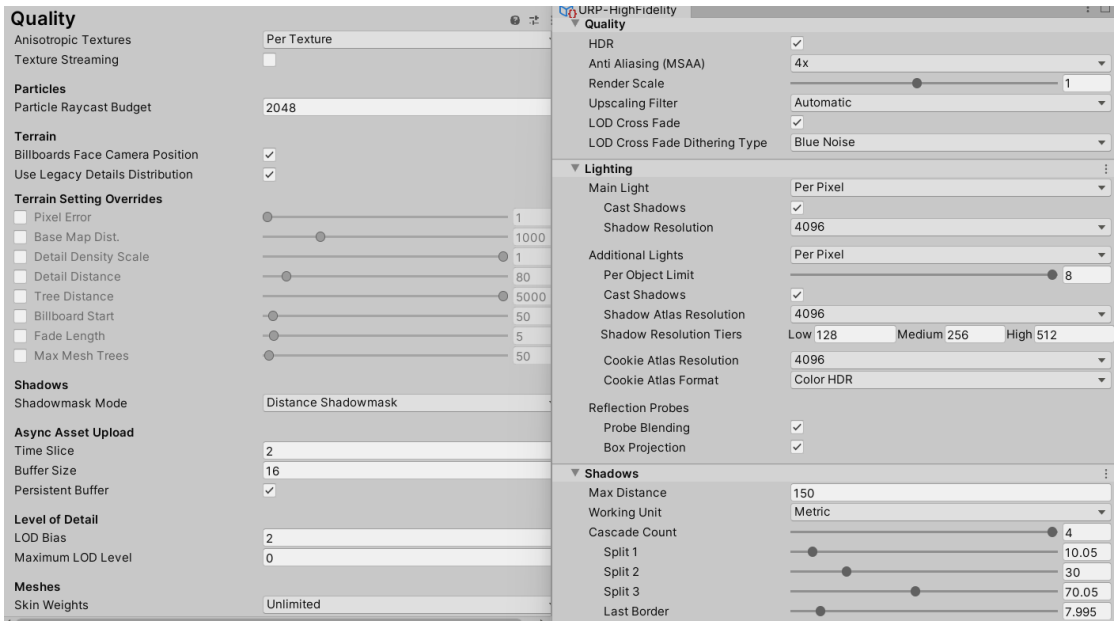


Figure 3.4: URP Parameter

Figure 3.5: Rendered view of CMM vs the actual machine model.

3.2.2 Unity's Assets Implemented

Unity is chosen as the core simulation platform because it integrates (i) real-time 3D rendering, (ii) physics simulation, (iii) event-driven interaction logic, and (iv) XR/VR functionality within a single development ecosystem. Specifically, Unity's built-in 3D physics system integrates the NVIDIA PhysX engine (the physics simulation engine that handles collisions and movement),

which provides mature rigid-body dynamics, collision detection, and ray-casting functionality needed to implement probe contact and collision feedback (Unity Asset Store, 2026).

From an XR perspective, Unity provides a standardized workflow through XR Plug-in Management and the OpenXR plug-in. XR Plug-in Management is used to configure XR providers for each build target and to validate XR project settings. OpenXR, an open industry standard that allows the software to run on different VR headsets without rewriting code for each one, is a royalty-free open standard designed to reduce platform-specific development by providing a common API layer across XR devices. In Unity, the OpenXR plug-in is configured via XR Plug-in Management, where the developer enables OpenXR on a per-target-platform basis and selects the required features and interaction profiles. Table 3.1 shows the development toolchain and Unity packages used, including SolidWorks, Unity, XR Interaction Toolkit, OpenXR plug-in, Input System, Unity UI, and PhysX-based physics.

3.2.3 Unity XR interaction stack and user input

The system employs Unity's modern XR and input stack:

- **XR Interaction Toolkit (XRIT):** XRIT is a high-level, component-based interaction framework for creating VR and AR experiences, built around Interactors, Interactables, and an Interaction Manager that mediates events between them. XRIT also includes locomotion components and tools for interaction feedback.

- **Unity Input System:** Input Actions separate the logical meaning of an input (e.g., "move," "select," "trigger") from the physical devices used to generate the input (e.g., controller thumb-stick, keyboard keys), enabling consistent interaction mappings between desktop and VR configurations.

- **Action enabling utilities:** XRIT provides helper extensions such as EnableDirectAction() and DisableDirectAction() for managing whether locally defined input actions are enabled by the owning MonoBehaviour (useful for modular scripts that manage their own input lifecycle). A summary of the software tools is provided in Table 3.1.

Table 3.1: Primary software components used in the VR CMM system

Component	Role in system	Key outputs
SolidWorks	Detailed CAD modeling of CMM and parts	Parametric CAD assemblies; exported meshes
Unity Engine	Rendering, physics, scripting runtime	Interactive 3D simulation; build artifacts
XR Plug-in Management + OpenXR	XR device configuration and runtime abstraction	Multi-device VR deployment configuration
XR Interaction Toolkit	High-level VR interactions and locomotion	Controller inputs, grab/point interactions
Unity Input System	Action-based input abstraction for VR/desktop	Device-agnostic movement and UI inputs
Unity UI (uGUI)	Menus, instruction panels, feedback display	Buttons, text, panels, runtime prompts

3.3 Hardware integration and deployment configurations

The VR training system is implemented and tested on a personal computer (specifications detailed in Chapter 5). This configuration provides sufficient rendering performance for viewing complex CAD-derived meshes while maintaining interactive frame rates. In practical testing, the Unity simulation project loads in approximately 20 seconds on this machine, enabling quick startup for classroom and lab training sessions.

The software architecture supports two primary deployment configurations:

1. **Desktop simulation mode (non-HMD):** Users navigate the virtual lab using keyboard and mouse controls in a first-person style. This configuration supports broad accessibility, since it does not require VR hardware.

2. **Immersive VR mode (HMD + controllers):** The Unity XR pipeline supports VR deployment via OpenXR. Unity's OpenXR plug-in aims to simplify cross-device targeting by enabling applications to build against OpenXR standards rather than device-specific APIs. Unity configures OpenXR via XR Plug-in Management, where OpenXR is enabled per platform build target and interaction profiles are selected.

Because Unity supports building for different target platforms via its Build Settings window, the same simulation project can be compiled and packaged for alternative deployment scenarios, including Windows desktop builds and Android-based standalone VR builds.

3.4 System architecture and user experience flow

3.4.1 Main menu and operational modes

The development begins with a main menu with three modes, as shown in Figure 3.6.

1. **Training Mode:** Provides guided, step-by-step instruction through UI tooltips and structured tasks. This mode is designed for novices and emphasizes procedural correctness (e.g., which step must occur before measurement).

2. **Operations Mode:** Removes step-by-step scaffolding and allows the trainee to use the CMM functions more independently, approximating how a user would operate the machine in a real lab once trained.

3. **Process Simulation Mode:** Demonstrates a complete "typical" CMM operation sequence automatically by animating and executing predefined actions (e.g., axis moves, probe touches, and computed results). This mode is primarily intended for demonstration, overview learning, and instructor-led explanation.

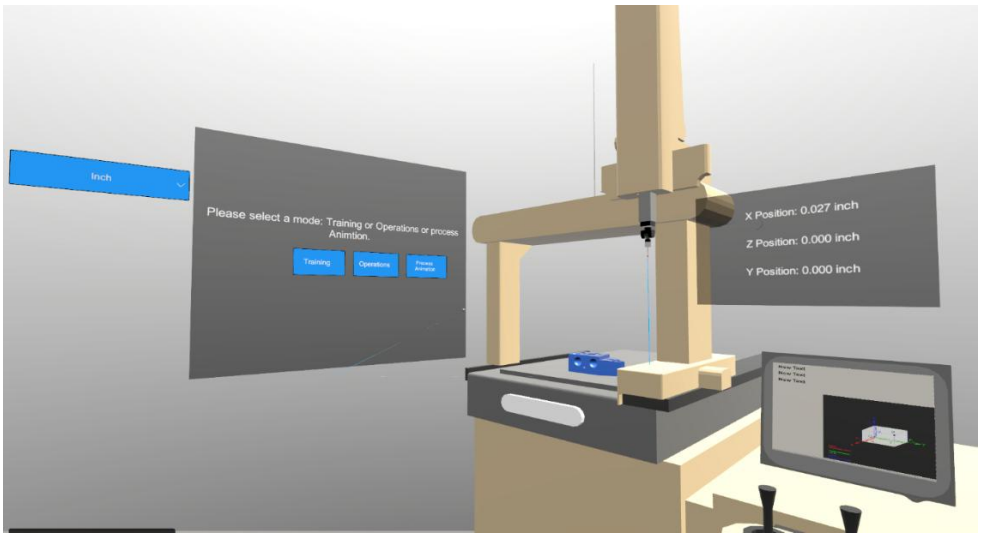


Figure 3.6: Main menu of the VR CMM application.

This top-level interaction structure is implemented as a state-driven UI controller. Central to this design is the CMMTrainingScript, which manages enabling and disabling UI elements, switching between modes, enabling or hiding step buttons, and maintaining the currentStepIndex when stepping through guided instruction. Unity UI elements, Buttons, Text fields, and panels, are organized under Unity's Canvas system, since Unity UI elements must exist within a Canvas GameObject to render properly.

3.4.2 Internal virtual display and instructor support

To enhance training usability, the simulator includes a virtual display screen inside the simulation environment, functioning as a control panel/monitor. This screen displays real-time probe coordinates, active coordinate system selection, and measurement mode indicators. The user can also select the unit system to work on (inches or millimeters). This design supports both self-guided learning (students view feedback directly) and instructor-led demonstrations (the virtual screen content can be observed externally if mirrored to a projector).

3.4.3 Real-time feedback cues

Real CMM operation includes explicit feedback that a probing event occurred (e.g., probe trigger signal). The VR system replicates this by providing immediate sensory feedback when the probe contacts the workpiece:

- A visual indicator (green light in the top-right corner) illuminates on successful probing contact.
- The system records the probe tip coordinates of contacts and prints them to the interface.
- An optional audio cue is played at collision events to reinforce that a measurement point is captured.

The audio feedback mechanism is implemented by adding an audio source to the probe or probe tip object, triggered on collision events.

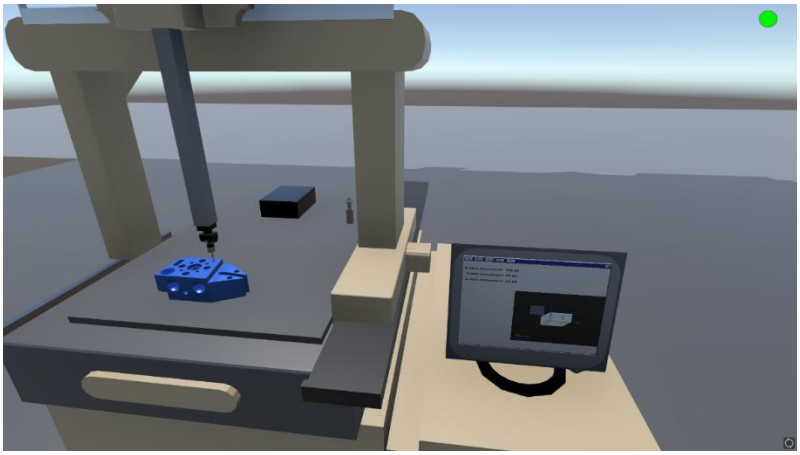


Figure 3.7: Collision feedback indicator (green light) and coordinate-capture event.

3.5 CMM modeling and interaction mechanics

3.5.1 CMM kinematics representation in Unity

A bridge-type CMM can be modeled as an orthogonal 3-axis machine, where the effective probe tip position is determined by translations along X, Y, and Z. In Unity, this is represented by structuring the machine as a hierarchy of transforms:

- An X-axis carriage (bridge or gantry motion)
- A Z-axis motion component (carriage slide)
- A Y-axis vertical motion component (spindle/quill motion carrying the probe head)

Let the machine's reference frame be {M}, and the probe tip position in Unity world coordinates be $\{p\}_w$. If the axes are modelled as independent translations (x, y, z) applied along orthogonal directions, then the probe tip position (relative to a machine datum origin $\{p\}_0$) can be simplified as:

$$p_w = p_0 + x\,i + y\,j + z\,k \qquad (3.1)$$

where i, j, and k are unit vectors corresponding to the machine axes. This direct Cartesian structure matches the physical motion of a standard bridge CMM and is straightforward to implement with Unity Transform translations.

3.5.2 User locomotion and movement input strategies

The system supports user navigation in the virtual lab environment and movement of machine components:

- **User navigation (VR locomotion**): The system includes an Action-based move provider script derived from XR input action patterns. In Unity XRIT, continuous locomotion is provided by components such as the Continuous Move Provider (Action-based), which moves the XR rig continuously based on 2D-axis input (typically a joystick thumb stick).
- **XR-Based Controller Movement Implementation:** The "ActionBasedMoveProvider" script reads a Vector2 input from left and right controller actions, sums these inputs, and applies translation to a target GameObject in the X–Z plane (world space). Enabling/disabling the actions follows XRIT's EnableDirectAction() and DisableDirectAction() pattern.
- **Machine axis control:** A dedicated script (JoystickMovement) processes joystick/controller inputs to translate specific machine GameObjects along their respective axes. The system

updates the UI with the current X, Y, Z positions, and records point coordinates upon probe collision.

To imitate physical CMM axis limits and avoid unrealistic "over-travel" in simulation, movement constraints are implemented using clamping logic. The RestrictMovement script stores an axis-specific original reference position and uses Unity's Mathf.Clamp() to constrain the current position to a maximum travel distance on a selected axis and direction. This corresponds to imposing a bounded region:

$$x \in [x_0, x_0 + x_{max}], \quad y \in [y_0, y_0 + y_{max}], \quad z \in [z_0, z_0 + z_{max}] \tag{3.2}$$

The maximum travel values are defined in inches and mapped to Unity units using a calibrated scaling factor. In addition, the script uses Gizmos to visualize travel limits in the Unity Scene view, supporting debugging and validation during development.

3.5.3 Collision reminder: physics and colliders

Probe contact detection is implemented using Unity's collision system, which relies on colliders and rigid bodies. Unity's built-in 3D collision detection and response are provided through its PhysX integration.

The probe tip is equipped with a collider, and the training workpiece is tagged and configured to respond to collision events. A collision event triggers coordinate capture and feedback, serving as the virtual analog of a probe trigger.

3.6 Implementation of CMM operational modules and mathematical formulation

The VR system implements a staged set of CMM operation modules that reflect standard practice: stylus qualification, coordinate system establishment, and feature measurement. These modules are implemented using C# scripts that manage collision-count logic, coordinate capture, coordinate transformations, and geometric fitting. The choice of "minimum viable measurement sets" (e.g., three points for a plane, three points for a circle) mirrors conventional geometric reconstruction strategies used in CMM programming when scanning is not required.

3.6.1 Stylus qualification module

Stylus qualification (probe calibration) is modeled as an interactive task where the user touches a reference sphere a specified number of times. The CollisionCountScript counts probe collisions with the calibration object and updates a UI countdown. Once the collision count reaches a defined threshold (five touches in the provided implementation), the system displays a completion message indicating successful qualification and a nominal stylus diameter value, as shown in Figure 3.8.

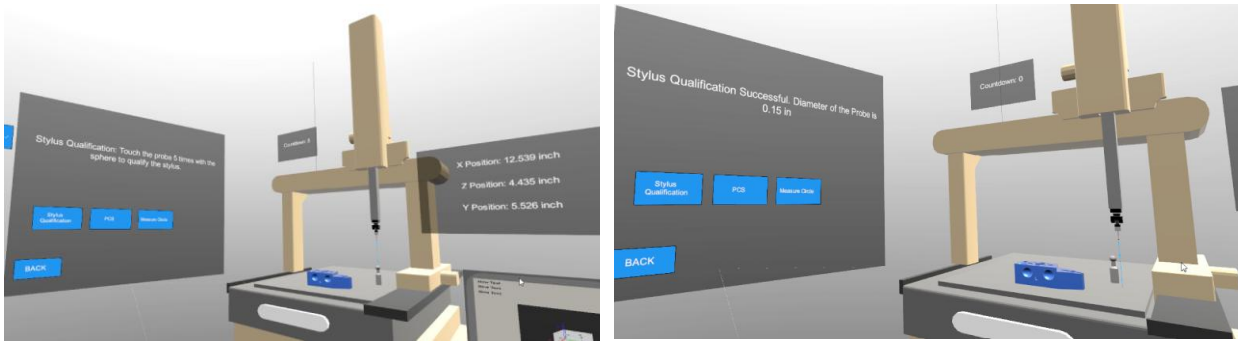


Figure 3.8: Stylus qualification training

Two mathematical ideas are embedded in this module:

1. **Contact event discretization:** Each collision represents a discrete probe hit event. The use of a cooldown (implemented by a coroutine delaying re-enabling collisions) prevents the physics engine from registering a continuous "contact" as multiple hits, thereby approximating discrete probe triggers.

2. **Diameter/scale inference:** The BallDiameter script demonstrates a method to infer a calibration sphere's diameter using mesh bounds. In Unity, Renderer.bounds.size.x gives the world-space dimension of the rendered object. For a sphere, this corresponds to the estimated diameter in Unity units. While this is not a full metrology-grade sphere-fitting procedure (which would typically involve multi-point sphere fitting), it provides a pragmatic basis for consistent scaling and instructional feedback.

3.6.2 Coordinate systems and part alignment module

CMM measurement requires that the MCS be related to the PCS. MCS is the machine's own internal map, and the PCS is a local map centered on the part being inspected. Establishing the PCS is like telling the machine to treat the corner of the part as a new origin. The simulator teaches this relationship by requiring the user to probe datum features in a guided sequence. The PCS module in the simulator is modelled after the 3-2-1 alignment principle: probing a primary plane, a secondary datum, and a tertiary datum to constrain freedoms and establish a stable reference frame. The importance of alignment is widely emphasized in metrology practice: the CMM cannot return meaningful dimensions until the part's reference frame is defined, and datums establish the origin and orientation used for all measurements.

The system's PartCoordinate script implements a staged alignment sequence as follows:

Stage 1 captures three touches on a primary plane, establishing a plane reference and constraining rotations.

Stage 2 captures two touches on a secondary plane/feature, constraining another rotation and one translation.

Stage 3 captures a tertiary point/plane, fixing the remaining translation, after which PCS is marked "Established."

In conceptual terms, a PCS is defined by:

- an origin o_p ,
- a set of orthonormal axes $\hat{u}_p, \hat{v}_p, \hat{w}_p$

Once defined, a point measured in machine coordinates p_m can be expressed in PCS coordinates p_p via:

$$p_p = R_{pm}(p_m - o_p) \qquad (3.3)$$

where R_{pm} is the rotation matrix mapping machine axes to part axes.

In the current implementation, the orientation step is simplified through guided probing stages and visual "guide" orientation updates, but the framework supports expanding to full-plane fitting and orthonormal frame computation.

3.6.3 Measurement modules and feature reconstruction

After stylus qualification and coordinate system definition, the system provides measurement workflows for fundamental geometric elements. The scripts capture probe contact points and compute derived geometry. This provides direct teaching value for trainees, showing how point probing becomes geometric construction.

3.6.3.1 Point measurement

A point measurement records probe tip coordinates at collision. In Unity, coordinate readouts are derived from Transform.position at the time of collision. When needed, the simulator converts these coordinates into the selected unit system.

3.6.3.2 Line measurement

A line can be defined by two points (p_1, p_2). The direction vector is:

$$d = p_2 - p_1 \tag{3.4}$$

and the parametric form is:

$$p(t) = p_1 + t\,d \tag{3.5}$$

The simplest trainee-facing outcome is often the distance between two measured points:

$$L = |p_2 - p_1| \tag{3.6}$$

which corresponds to a point-to-point measurement.

3.6.3.3 Plane measurement

A plane can be defined by three non-collinear points p_1, p_2, p_3. The plane normal is:

$$n = (p_2 - p_1) \times (p_3 - p_1) \tag{3.7}$$

where 'p' represents the coordinates of the points and the plane equation is:

$$n \cdot (p - p_0) = 0 \qquad (3.8)$$

where $p_0$denotes a reference point on the plane and can be chosen as any of the points p_1, p_2,or p_3.The simulator's three-point plane workflow provides learners an intuitive sense of how multiple touches establish surface orientation.

3.6.3.4 Circle measurement (three-point reconstruction)

The implemented circle routine records three points and computes the circle center and radius. The provided script computes the circle in a plane (treated as ZX-plane in the implementation). The method is based on intersecting perpendicular bisectors:

1. Compute the midpoints $m_{12}$and m_{23}as:

$$m_{12} = \frac{p_1+p_2}{2}, \quad m_{23} = \frac{p_2+p_3}{2} \qquad (3.9)$$

   and determine the perpendicular directions to the chord segments in the reference plane.

2. Solve for the intersection point (circle center). The script checks collinearity using a determinant threshold; if the determinant is near zero, points are collinear (no circle exists).

Finally, radius is computed as the distance from center to a point:

$$R = |c - p_1| \qquad (3.10)$$

This approach is well-suited for training because it is interpretable: trainees can conceptually connect "three points determine a circle" to the computed result. The following Table 3.2 summarizes the geometric feature types implemented in the VR CMM system, along with the number of probe points required in the training version and the primary computations used to derive each feature.

Table 3.2: Feature computations used in the VR CMM system

Feature type	Probe points used (training version)	Primary computation
Point	1	Record (x, y, z) at contact

Feature type	Probe points used (training version)	Primary computation
Line	2	$(d = p_2 - p_1, \quad L = \|p_2 - p_1\|)$
Plane	3	Normal $(n = (p_2 - p_1) \times (p_3 - p_1)$
Circle	3	Perpendicular bisectors → center (c), radius (R)

3.6.4 Unit conversion and scaling between Unity and engineering units

A practical issue in simulation-based metrology training is maintaining consistent mapping between simulated positions and engineering units. The simulator implements a mapping strategy where travel limits are specified in inches (e.g., 12 in, 15 in, 18.5 in), and positions are translated into Unity-world constraints using a calibration factor. In the scripts, a scaling parameter unityToInch is estimated from a chosen reference travel and adjusted by a ball radius term, and its inverse InchTounity is used to convert measured Unity positions into inches for display.

In conceptual terms, define a scale (S) such that:

$$S = \frac{\Delta u}{\Delta inch} \qquad (3.11)$$

where (Δu) is the Unity-coordinate travel corresponding to a known physical travel $(\Delta inch)$. Then:

$$inch = \frac{u}{S}, \quad u = S \cdot inch \qquad (3.12)$$

The user interface includes a unit dropdown in the axis movement script to switch between inches and millimeters, applying the standard conversion: $1 \text{ in } = 25.4 \text{ mm}$.

3.6.5 Raycast-based distance demonstrations

The DistanceMapper script uses raycasting to compute distances from the tool origin to the first intersection point on the workpiece along selected axes. In Unity, Physics.Raycast casts a ray and returns hit information, including the collision point in world space.

This is used for demonstrations of directional clearance checks and simplified distance mapping tasks. The distance along a chosen axis is remembered as:

$$D_x = |x_0 - x_{hit}|, \quad D_y = |y_0 - y_{hit}|, \quad D_z = |z_0 - z_{hit}| \qquad (3.13)$$

This reinforces the trainee's understanding that many CMM measurements reduce axis-referenced differences in a defined coordinate frame.

3.7 Process simulation mode and automated demonstration sequences

A distinguishing feature of the simulator is the process simulation mode, which autonomously demonstrates CMM operations. Rather than relying on the trainee to execute each motion manually, this mode runs predefined sequences that move machine components to target positions in the correct procedural order and triggers instructional messages.

3.7.1 Animation and sequencing mechanism

The automation is implemented with a SequentialMover script. Each automated step is represented as a move sequence specifying as follows:

- a list of objects to move,
- a target position,
- a delay before movement,
- a movement duration.

Movement is interpolated using linear interpolation:

$$p(t) = (1 - t)\, p_0 + p_1 \qquad (3.14)$$

where (t) progresses from 0 to 1 during the movement duration. The system organizes moves into groups so that multiple objects can be moved simultaneously (e.g., coordinated-axis motions), and uses coroutines to schedule delays and to wait for moves to complete. Finally, an optional "keep at target position" coroutine ensures the objects remain at the target positions during the

demonstration mode. Table 3.3 provides a summary of the script used in the VR model development.

Table 3.3: Summary of Scripts and Their Role

Script / Class Name	**Role in the VR CMM System**
CMMTrainingScript	Central state-driven UI controller. Manages enabling/disabling UI elements, switching between Training, Operations, and Process Simulation modes, toggling step buttons, and tracking currentStepIndex during guided instruction.
ActionBasedMoveProvider	Handles VR user locomotion. Reads Vector2 input from left and right controller actions, sums them, and translates the XR rig in the X–Z world plane. Uses XRIT's EnableDirectAction() / DisableDirectAction() pattern.
JoystickMovement	Controls CMM machine axis movement. Processes joystick/controller inputs to translate specific machine GameObjects along their respective axes (X, Y, Z), updates the UI with current positions, and records point coordinates on probe collision.
RestrictMovement	Enforces CMM axis travel limits. Stores an axis-specific original reference position and uses Mathf.Clamp() to constrain movement within bounded ranges. Draws Gizmos in the Scene view for debugging travel limits.
CollisionCountScript	Implements the stylus qualification module. Counts probe-to-calibration-sphere collisions, updates a UI countdown, and displays a completion message with nominal stylus diameter once the threshold (5 touches) is reached. Uses a cooldown coroutine to prevent multi-registration.
BallDiameter	Infers the calibration sphere's diameter from Unity mesh bounds (Renderer.bounds.size.x). Provides a pragmatic scaling/calibration reference for instructional feedback during stylus qualification.
PartCoordinate	Implements the Part Coordinate System (PCS) establishment module using 3-2-1 alignment. Manages a three-stage probing sequence (primary plane → secondary

	datum → tertiary datum) and marks the PCS as "Established" once all stages complete.
DistanceMapper	Uses Physics.Raycast to compute distances from the tool origin to the first workpiece intersection along selected axes. Supports directional clearance checks and simplified distance-mapping demonstrations.
SequentialMover	Drives the Process Simulation mode. Executes predefined automated move sequences (target position, delay, duration) using linear interpolation. Organizes moves into groups for simultaneous axis motion, uses coroutines for scheduling, and optionally holds objects at target positions.

3.8 Chapter Summary

The system implements core CMM operational modules: stylus qualification, PCS establishment consistent with common datum-alignment logic, and feature measurement (point, line, plane, circle), and presents results through immediate UI feedback and a virtual display screen. Measurement algorithms rely on fundamental geometric computations such as cross-products for planes, and perpendicular bisector intersection for circles, reinforcing the mathematical basis of coordinate metrology as part of the training experience. Process Simulation extends the system by demonstrating complete operational sequences automatically via scripted movement interpolation.

Chapter 4 builds upon this simulator as an execution environment for automated inspection planning, focusing on optimized probe path generation and sequence planning (A*, TSP framing, and 2-opt refinements) using the virtual CMM as a safe verification platform.

Chapter 4. Path Planning

4.1 Overview of Path Planning in CMM

In modern manufacturing, automating the inspection path planning for Coordinate Measuring Machines (CMMs) is crucial to improving efficiency and reducing reliance on manual programming. A CMM probe must visit different feature locations, such as holes, slots, surfaces, cones and other geometric primitives, on a workpiece in measurements. Determining an optimal and collision-free path for the probe to travel between these feature points is a complex task that has traditionally required significant human effort and expertise. Efficient inspection path planning for multiple features has been identified as a key task for CMM applications. Manual path programming can incur high labour costs and long process cycles.

CMM path planning encompasses two interrelated problems. As introduced in Section 2.2.2, the first is a sequence optimization problem formulated as a Open Travelling Salesman Problem (TSP), where features are treated as cities that must be visited once in an order that minimizes travel cost. The second problem is a motion planning challenge: generating a collision-free trajectory for the probe to follow along the optimized sequence, avoiding collisions with the workpiece geometry, fixtures, or user-defined obstacles. Both are critical as an inefficient sequence wastes time with unnecessary movements, while a poor trajectory could cause the probe to strike the part, risking damage to both the probe and the workpiece.

This chapter details the implementation of both and presents a comprehensive approach to CMM path planning implemented in a Python-based virtual environment with a graphical user interface (GUI). The approach integrates computer vision for automatic feature detection, combinatorial optimization for measurement sequence planning, shape-specific waypoint generation for probe contact points, and a graph-based A* search algorithm for collision-free path planning. Critically, this chapter introduces a complete collision-avoidance framework that enables the operator to define rectangular obstacle regions on the workpiece image and ensures the generated path navigates safely around them while remaining within the workpiece's outer contour. The system architecture, algorithms, and demonstration of the complete system on a representative workpiece are presented in the following sections.

4.2 System Architecture and Software Framework

The path planning solution is developed and tested in a Python-based virtual environment that simulates the CMM workspace through a desktop graphical user interface (GUI) built with the PyQt5 framework. The architecture is modular, consisting of several processing stages that mirror a real CMM inspection planning pipeline. Figure 4.1 illustrates the complete data flow from image input to executable probe coordinates. The process comprises seven stages, each implemented as an independent module that can be improved or replaced without affecting the others.

4.2.1 Image Input and Preprocessing

The input to the path planning pipeline is a two-dimensional image of the workpiece, captured as a top-down photograph or CAD rendering representing the part as it would appear on the CMM table. In this work, the input image has dimensions of 864 × 542 pixels, where each pixel corresponds to the smallest addressable unit of the image grid (Figure 4.1(a)). Using the OpenCV library, the image is converted into grayscale, and a binary threshold is applied to isolate geometric features as distinct white regions against a dark background (Figure 4.1(b)). This produces a clean binary image suitable for contour-based feature detection.

Because all subsequent processing is performed by OpenCV, which operates directly on the pixel grid of the image, every output of the detection stage feature center coordinates, radii, areas, and contour points is expressed in pixel units. In this coordinate system, the origin (0, 0) is located at the top-left corner of the image, the x-axis increases to the right (reaching a maximum of 863), and the y-axis increases downward (reaching a maximum of 541). When OpenCV functions such asRS, cv2.minEnclosingCircle, or cv2.moments are applied to the binary image, they return geometric properties measured in pixels. These pixel-based values are the raw detection output that populates values in inches after scaling.

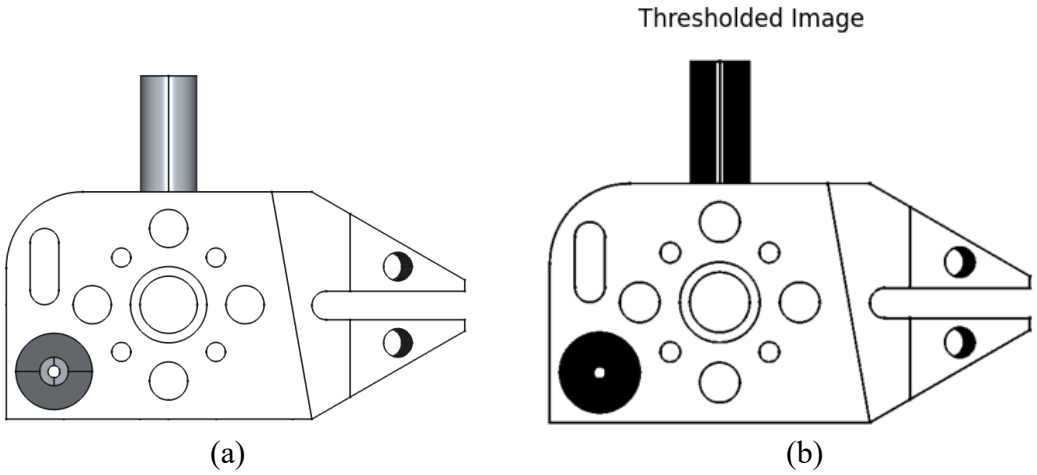


Figure 4.1: Conversion of Input Image to Threshold Image

4.2.2 Feature Detection and Extraction

Computer vision algorithms trace the boundaries (contours) of each white region in the binary image. The geometric properties, such as centroids, areas, perimeters, bounding rectangles, enclosing circles, and fitted ellipses, are abstracted for each detected feature. Noise and duplicate detections are filtered out using area thresholds and the Intersection over Union (IoU) metric. The result is a structured table of genuine measurement features with their locations and dimensions (Figure 4.2).

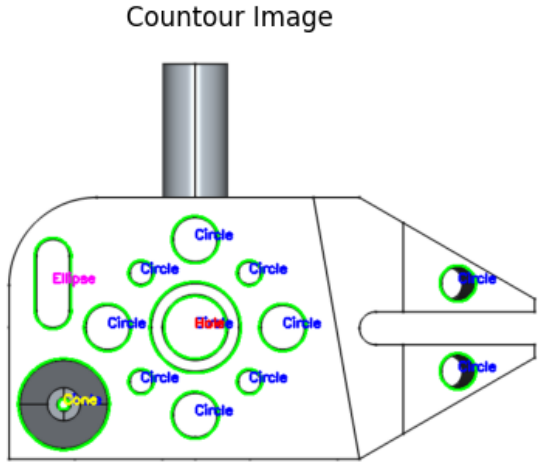


Figure 4.2: Detected feature contours overlaid on the original image

4.2.3 Feature Classification and Grouping

Each detected contour is classified by geometric types (circle, slot, rectangle, cone, or polygon) using circularity metrics, polygon vertex count, and ellipse axis-ratio analysis. Similar features are grouped for organized visualization and to inform measurement strategy (Figure 4.3).

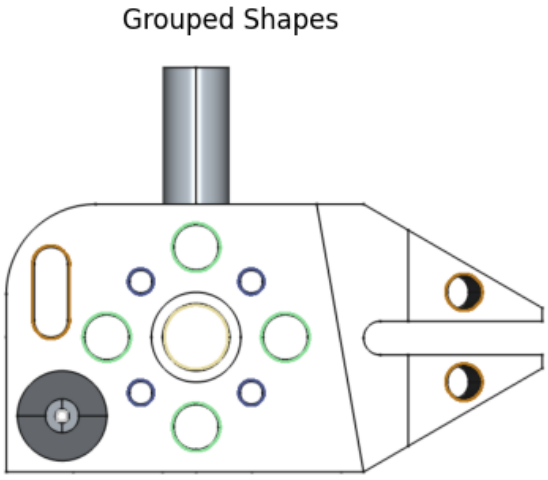


Figure 4.3: Detected features grouped by type

4.2.4 Sequence Optimization (TSP Solver)

A complete graph is constructed with each feature centre as a node and edge weights representing Euclidean distances. A nearest-neighbour heuristic provides an initial tour, which is subsequently refined using the 2-opt local search algorithm to eliminate crossing paths and reduce the total travel distance (Figure 4.4).

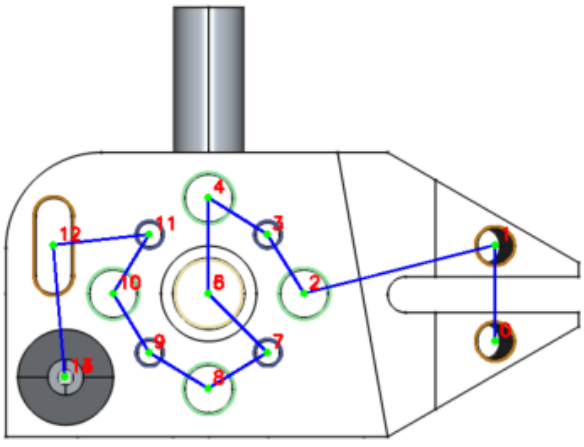


Figure 4.4: Optimized measurement sequence connecting all features

For each feature in the optimized sequence, shape-specific measurement waypoints are generated as seen in Figure 4.5. Circles receive three equally-spaced contact points around the circumference; slots receive six parametric contact points; rectangles receive diagonal corner points; and general polygons are approximated using the Douglas–Peucker algorithm (explained in Section 4.8.4).

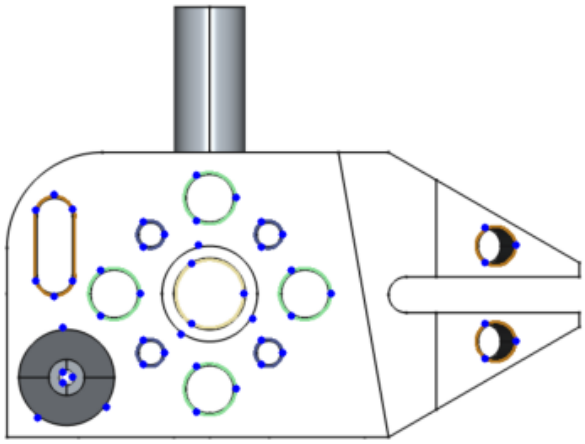


Figure 4.5: Waypoint generation on a grid

4.2.5 Collision Avoidance and Obstacle-Aware Path Planning

The operator can interactively define rectangular obstacle bounding boxes on the workpiece image via the GUI. The system adjusts waypoints that fall within obstacles, filters out features whose centres lie within obstacles, and employs the A* graph search algorithm on a visibility graph to find the shortest collision-free path between consecutive features. Figure 4.6 shows the final path connecting all the waypoints.

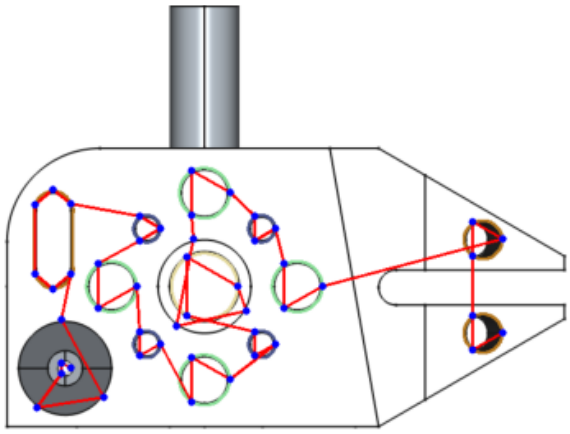


Figure 4.6: Visualization of A pathfinding on the grid*

4.2.6 Coordinate Transformation and Output

All path coordinates are transformed from image pixel coordinates (top-left origin) to a bottom-left Cartesian coordinate system scaled by a user-specified factor (pixels to physical units). The final output is a list of scaled waypoint coordinates suitable for CMM execution. The GUI application, implemented as the CMMPathPlannerApp class, provides the operator with an interactive workflow: browsing and loading the workpiece image, setting the coordinate scale factor, drawing obstacle bounding boxes directly on the image, initiating the processing pipeline, and visualizing intermediate results (TSP path, waypoints, final collision-free path). Processing runs in the background so the app doesn't freeze while the path is being computed and after processing, it displays the path overlaid on the workpiece image.

Table 4.1 lists the software libraries used throughout the path planning system and their roles. The following sections describe each stage in detail, beginning with the shape detection (Section 4.3) and proceeding through classification (Section 4.4), sequence optimization (Section 4.5), waypoint generation (Section 4.6), collision avoidance (Section 4.7), and the A* pathfinding algorithm (Section 4.8).

Table 4.1: Image Preprocessing Pipeline

Panel	**What to show**	**How to generate**
(a) Original input	Color image of workpiece	Image loaded using OpenCV cv2.imread (path)
(b) Grayscale	Single-channel intensity	Colour-to-grayscale conversion cv2.cvtColor (img, COLOR_BGR2GRAY)
(c) Binary threshold	Black/white with features isolated	Pixels above 250 → white; rest → black cv2.threshold (gray, 250, 255, THRESH_BINARY)
(d) Detected contours	Contours drawn in green on the original image	Contour tracing on the binary image cv2.drawContours (img, contours, -1, (0, 255, 0), 2)

4.3 Shape Detection and Preprocessing

Shape detection is the first step in the path planning. It identifies the geometric features (holes, circles, slots, etc.) on the workpiece that the CMM probe must visit. The system uses the open-source computer vision library OpenCV to process an image of the workpiece and automatically locate these features (see Appendix A for the OpenCV functions used). The entire process consists of the following four stages: loading the image, simplifying it, isolating its features, and extracting their geometric properties.

Stage 1: Grayscale Conversion. The input image, either a photograph of the physical part or a rendered view of its CAD model, is first converted from colour to grayscale. A colour image stores

three values per pixel (red, green, and blue intensities), but shape detection only needs brightness information. Grayscale conversion collapses these three channels into a single intensity value per pixel, simplifying subsequent processing without losing the edge and shape information needed for detection (Figure 4.8).

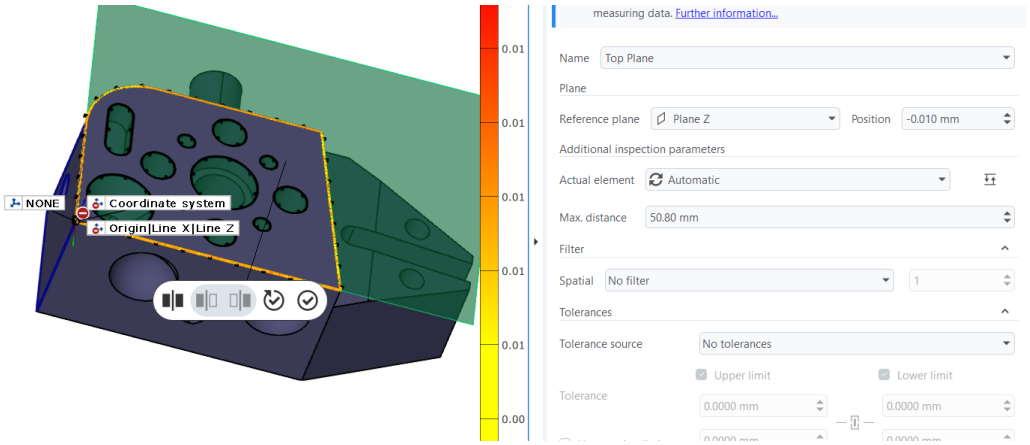


Figure 4.7: Top place of CAD model of workpiece

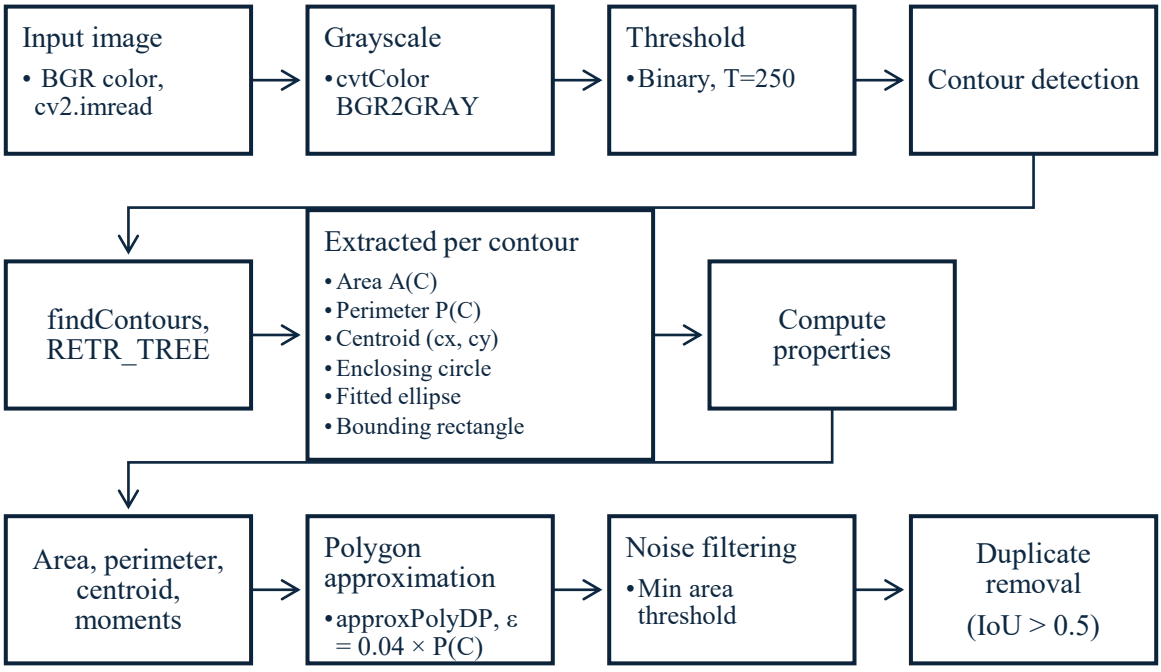


Figure 4.8: Shape Processing

Stage 2: Binary Thresholding. Next, the grayscale image is converted into a strict black-and-white image using a technique called binary thresholding. Every pixel whose brightness exceeds a cutoff value of 250 (on a 0–255 scale) is set to pure white; everything else becomes pure black. This produces a clean separation between the workpiece features (white regions) and the background (black). A high threshold of 250 works well here because engineering drawings and CAD renderings have sharp, high-contrast edges, and features appear very bright against a dark background.

Stage 3: Contour Detection. With the features isolated as white regions, the system traces the boundary of each region. These boundaries are called contours. A contour is simply the outline curve that separates a feature from the background, much like tracing around a shape with a pencil. The detection algorithm captures all contours, including nested ones (e.g., a smaller circle within a larger circle, representing a cone or a countersunk hole). To save memory, straight portions of each boundary are stored only as their start and end points, rather than every pixel along the edge.

Stage 4: Geometric Property Extraction. For each detected contour, the system computes several measurements that will later be used to classify the feature type and determine where the CMM probe should go. These include the area (the size of the enclosed region), the perimeter (the total length of the boundary), and the centroid (the geometric centre, calculated from weighted averages of the boundary coordinates called image moments). The system also fits standard geometric shapes to each contour: the smallest enclosing circle (useful for identifying round features), the best-fit ellipse (useful for elongated features like slots, providing the major and minor axis lengths and rotation angle), and the tightest oriented bounding rectangle.

Each contour is then simplified using the Douglas–Peucker (RDP) algorithm, which reduces the number of boundary points while preserving the shape's essential corners and curves, as shown in Figure 4.12. The algorithm works by recursively removing points that lie within a small tolerance ($\varepsilon = 4\%$ of the contour's perimeter) of a straight-line approximation. If a point deviates only slightly from the line connecting its neighbours, it is discarded. The result is a clean polygon with far fewer vertices that still faithfully represents the original shape, making the subsequent classification step faster and more reliable.

Finally, the system removes detections that are noise or duplicates. Very small contours below a minimum area threshold are discarded, as these typically correspond to image artefacts rather than real features. Duplicate detections where two contours trace essentially the same feature are identified using the Intersection over Union (IoU) metric. IoU measures how much two shapes overlap: it divides the area of their overlap by the area of their combined coverage.

$$IoU(C^1, C^2) = \frac{|M^1 \cap M^2|}{|M^1 \cup M^2|} \qquad (4.1)$$

An IoU value of 1.0 means the two shapes are identical; a value of 0 means no overlap. If two contours have an IoU greater than 0.5, meaning they overlap by more than half, the duplicate is removed. The surviving contours represent the workpiece's true measurement features, each fully described by its location, size, shape parameters, and simplified boundary, ready for sequencing and path planning.

4.4 Feature Classification and Grouping

4.4.1 Shape Classification

Each detected contour is classified into a geometric type based on its circularity, the number of vertices in its polygon approximation, and the ellipse axis ratios. The circularity metric is computed as:

$$Circularity = 4\pi \cdot \frac{A(C)}{P(C)^2} \qquad (4.2)$$

where A(c) represents the area and P(c) is the perimeter of the circle. A perfect circle has a circularity of 1.0. The classification rules applied in the implementation are as follows. If the circularity exceeds 0.85, the contour is classified as a circle. For contours with lower circularity, the axis ratio of the fitted ellipse is examined: if the ratio of the minor axis to the major axis (ma/MA) falls between 0.4 and 0.6, the contour is classified as a slot (elongated elliptical feature). Contours whose polygon approximation yields exactly four vertices are classified as rectangles, and those with three vertices as triangles. Contours with five or more vertices that do not meet the circularity or slot criteria are classified as general polygons.

4.4.2 Circle Relationship Analysis (Cone Detection)

For nested circle pairs, common in features such as countersunk holes or cones, the system performs a pairwise analysis of detected circles. The center distance between two circles with centers (x_1, y_1) and (x_2, y_2) is computed as the Euclidean distance. If the center distance is less than 0.5 pixels (indicating near-concentric circles) and the inner radius r_2 is less than 0.6 times the outer radius r_1, the pair is classified as a cone. If $0.6r_1 \leq r_2 < 0.8r_1$, the pair is classified as concentric circles. This hierarchical classification ensures that complex multi-contour features are correctly identified as single measurement entities.

4.4.3 Feature Grouping

After classification, features of the same type are grouped based on geometric similarity metrics. For circles, grouping is performed by comparing radii and areas: two circles are placed in the same group if their radius difference $\Delta r = |r_i - r_j| \leq \tau_r = 5$ pixels and their area difference $\Delta A = |A_i - A_j| \leq \tau_a = 20$ pixels². For slots, grouping is based on aspect ratio similarity: the aspect ratio $R_i = MA_i / ma_i$ is computed, and two slots are grouped together if $|\Delta R| = |R_i - R_j| \leq 0.1$ and their area difference is within 20 pixels². Similar features are grouped in the same color as shown in Figure 4.3. This grouping organizes the detected features for visualization and can inform the measurement strategy, as features within the same group often share similar measurement procedures. Table 4.2 lists the classification criteria and the grouping metric for all the features.

Table 4.2: Shape Classification Rules

Shape Type	**Classification Criterion**	**Grouping Metric**
Circle	Circularity > 0.85	Radius difference ≤ 5 px, area difference ≤ 20 px²
Slot	Ellipse axis ratio (ma/MA) between 0.4 and 0.6	Aspect ratio difference ≤ 0.1, area difference ≤ 20 px²

Rectangle	Polygon approximation has exactly 4 vertices	Area and dimension similarity
Triangle	Polygon approximation has exactly 3 vertices	Area similarity
Cone	Concentric circles with $r_2 < 0.6 \times r_1$	Outer radius and area
Polygon	Polygon approximation has ≥ 5 vertices, low circularity	Vertex count and area

4.5 TSP-Based Sequence Optimization

With all feature coordinates and types identified, the system determines an efficient sequence to visit them. This is formulated as an open Traveling Salesman Problem (TSP) to find a permutation τ of the n features that minimizes the total Euclidean travel distance. For n feature centers $c_i = (x_i, y_i)$ for i = 1, 2, …, n, the objective is to minimize:

$$L(\tau) = \sum_{i=1}^{n-1} d\left(c_{\tau_i}, c_{\tau_{i+1}}\right) + d\left(c_{\tau_n}, c_{\tau_1}\right) \quad (4.3)$$

where τ represents the visiting sequence (tour), c_{τ_i} denotes the i-th node in the sequence, and $d(p, q)$ is the distance between two nodes. The final term closes the tour by returning to the starting node. The TSP is NP-hard, meaning that finding the globally optimal solution by exhaustive enumeration is computationally infeasible for large n. The implementation therefore employs a two-stage heuristic approach: the nearest-neighbor heuristic for initial tour construction, followed by 2-opt local search for iterative improvement. Algorithm 4.1 presents the combined pseudocode for the TSP-based sequence optimization, integrating the nearest-neighbor construction heuristic with the 2-opt local improvement procedure and Figure 4.9 represents the initial TSP path.

Algorithm 4.1: TSP Sequence Optimization (Nearest Neighbor + 2-Opt)

```
Input: Feature centers {c_1, c_2, ..., c_n}
Output: Optimized visitation order
// Phase 1: Nearest-Neighbor Construction
visited = {c_1}; tour = [c_1]; current = c_1
FOR k = 2 TO n:
  next = argmin d(current, c_j) for c_j not in visited
  tour.append(next); visited.add(next); current = next
// Phase 2: 2-Opt Improvement
improved = True
WHILE improved:
  improved = False
  FOR i = 0 TO n-2:
    FOR j = i+2 TO n-1:
      delta = d(tour[i], tour[i+1]) + d(tour[j], tour[j+1])
            - d(tour[i], tour[j]) - d(tour[i+1], tour[j+1])
      IF delta > 0:
        Reverse tour[i+1 .. j]
        improved = True
RETURN tour
```

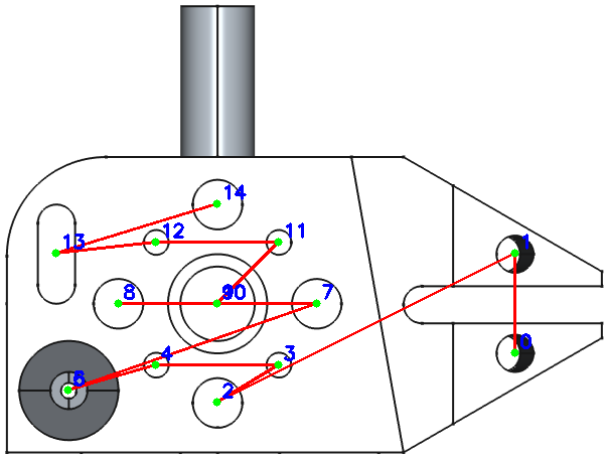


Figure 4.9: Initial TSP Path

4.5.1 Nearest-Neighbor Heuristic

The nearest-neighbor heuristic builds an initial inspection path using a greedy strategy. It starts from an initial feature (usually the first detected point) and then repeatedly moves to the closest feature that has not yet been visited. At each step, the algorithm selects the nearest unvisited point and continues until all features are covered. The total path length is computed as:

$$L_{nn} = \sum_{k=1}^{n-1} d(v_{k-1}, v_k) \qquad (4.4)$$

where:

- L_{nn}= total length of the nearest-neighbor path,
- n= total number of features (measurement points),
- v_k= the k-th selected feature in the path,
- $d(v_{k-1}, v_k)$= distance between two consecutive features $v_{k-1}$and v_k.

This equation simply sums the distances between consecutive points in the generated path. The time complexity describes how an algorithm's running time increases as the input size (n) grows. The time complexity of the nearest-neighbor heuristic is $O(n^2)$. This is because:

- At each step, the algorithm must check all remaining unvisited points to find the closest one.
- Initially, there are n points, then $n-1$, then $n-2$, and so on.

So the total number of comparisons becomes:

$$n + (n-1) + (n-2) + \cdots + 1 \approx \frac{n(n-1)}{2} \qquad (4.5)$$

It grows proportionally to n^2, hence the complexity is quadratic**.** Although the nearest-neighbor heuristic does not guarantee the optimal path, it is computationally efficient and easy to implement. In practice, it provides a good initial solution that can be further improved using local optimization techniques such as the 2-opt algorithm.

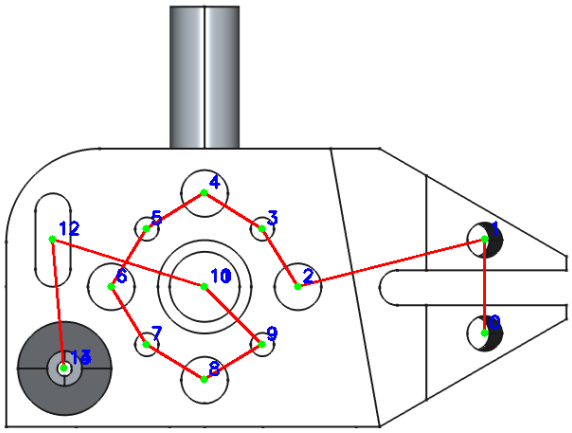


Figure 4.10: Path Using Nearest Neighbor Heuristic

4.5.2 2-Opt Local Improvement

The 2-opt algorithm is employed as a local search technique to improve the initial inspection path generated by the nearest-neighbor heuristic. While the nearest-neighbor method provides a feasible solution with low computational cost, it often produces paths containing inefficiencies such as edge crossings and unnecessary detours. The 2-opt algorithm addresses these issues by iteratively refining the path. The core idea of the 2-opt method is to examine pairs of non-adjacent edges in the current tour and determine whether exchanging them reduces the total path length. For two edges $(v_i, v_{i+1})$and (v_j, v_{j+1}), where $i < j$, the potential improvement is evaluated using:

$$\Delta L = d(v_i, v_{i+1}) + d(v_j, v_{j+1}) - d(v_i, v_j) - d(v_{i+1}, v_{j+1}) \qquad (4.6)$$

If $\Delta L > 0$, the swap results in a shorter path, and the segment between $v_{i+1}$and $v_j$is reversed. This operation effectively removes path crossings and reduces unnecessary travel distance. As implemented in the proposed algorithm, the initial tour is constructed using the nearest-neighbor approach:

$$next = arg\,min\, d(current, c_j) \qquad (4.7)$$

This provides a starting sequence of feature centers $\{c_1, c_2, \ldots, c_n\}$. Subsequently, the 2-opt procedure iteratively refines this sequence by evaluating all valid edge pairs as shown in the algorithm logic below. The algorithm continues until no further improvement is possible, at which point a locally optimal solution is obtained.

The computational complexity of the 2-opt algorithm is $O(n^2)$ per iteration, as all valid pairs of edges must be evaluated. However, in practice, the number of iterations required for convergence is relatively small, making it efficient for moderate problem sizes typically encountered in CMM inspection planning.

In the context of CMM applications, this refinement step significantly improves probe path efficiency by reducing unnecessary motion, minimizing traversal distance, and eliminating inefficient probe movements. The combined use of nearest-neighbor initialization and 2-opt refinement yields high-quality inspection paths that are typically close to optimal for feature sets ranging from tens to hundreds of measurement points.

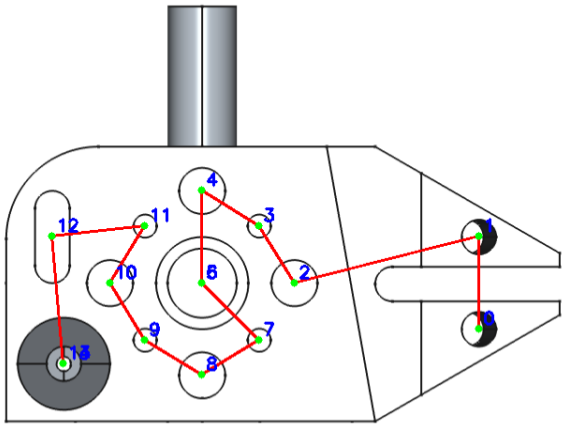


Figure 4.11: Path Using 2-opt

4.6 Waypoint Generation

After determining the optimal feature sequence, the system generates measurement waypoints for each feature. These waypoints represent the specific probe contact points required to measure each geometric feature. The waypoint generation strategy is shape-dependent, ensuring that each feature type receives an appropriate set of measurement points based on its geometric properties.

4.6.1 Circle Waypoints

For a circle with center (x_c, y_c) and radius r, three equally spaced waypoints are generated around the circumference at angular positions:

$$\theta_k = 2\pi k/3 \; where \;\; k \in 0,1,2 \qquad (4.8)$$

The waypoint coordinates are computed as:

$$w_k = (xc + r \cdot cos\theta_k, yc + r \cdot sin\theta_k) \qquad (4.9)$$

Three points are the minimum required to define a circle, making this an efficient measurement strategy. Equal angular spacing maximizes the geometric sensitivity of the circle fit, thereby minimizing uncertainty in the computed center and radius.

4.6.2 Slot (Ellipse) Waypoints

For a slot modelled as an ellipse with center (x_c, y_c), semi-major axis a = MA/2, semi-minor axis b = ma/2, and rotation angle α, six waypoints are generated at the angular positions:

$$\theta_k \in 45°, 90°, 135°, 225°, 270°, 315° \qquad (4.10)$$

The parametric coordinates are computed as:

$$X_k = xc + a \cdot cos\theta_k \cdot cos\alpha - b \cdot sin\theta_k \cdot sin\alpha \qquad (4.11)$$

$$Y_k = yc + a \cdot cos\theta_k \cdot sin\alpha + b \cdot sin\theta_k \cdot cos\alpha \qquad (4.12)$$

where R(α) is the 2×2 rotation matrix. The six contact points are distributed to adequately characterize the elongated shape, with points concentrated near the ends and sides of the slot.

4.6.3 Plane Waypoints

Planes are treated as approximate rectangles, and their waypoints are placed at the two diagonal corners of their minimum-area bounding rectangle. Given the four corner points {p_1, p_2, p_3, p_4} obtained from minAreaRect, the waypoints are $w_1 = p_1$ and $w_2 = p_3$ (opposite corners). This provides the two points needed to verify the rectangle's diagonal dimension and position.

4.6.4 Polygon Waypoints

For general polygonal features, the Douglas–Peucker algorithm is used to simplify the boundary contour into a reduced set of representative points. This well-established curve simplification technique reduces the number of vertices in a polygon while preserving its overall shape (Zygmunt & Róg, 2026).

The algorithm operates by recursively eliminating points within a specified tolerance distance of a line segment. It begins by connecting the start and end points of a curve segment with a straight line, then identifies the point with the maximum perpendicular distance from this line. If this distance exceeds a predefined tolerance, the point is retained and the process is applied recursively to the resulting sub-segments, as illustrated in Figure 4.12**.** (a) The dashed white polyline with gray vertices represents the original, densely sampled contour prior to simplification. (b) In the top-right panel, the algorithm initializes by connecting the two endpoints and identifies the farthest perpendicular point from that segment; the green approximation remains coarse, retaining only a few critical vertices. (c) The bottom-left panel depicts an intermediate iteration in which additional vertices exceeding the distance tolerance have been recursively incorporated, yielding a closer but still simplified approximation. (d) The bottom-right panel shows the near-final result, where the simplified polyline closely tracks the original contour, retaining vertices at sharp curvature changes while discarding redundant points along straighter segments. Otherwise, intermediate points are discarded, ensuring that only the most geometrically significant points are preserved. In this work, the approximation tolerance is defined as:

$$\varepsilon = 0.04 \times P(C) \qquad (4.13)$$

where $P(C)$ denotes the perimeter of the polygon. This tolerance value controls the level of simplification. A larger ε produces fewer waypoints and a more simplified contour, while a smaller ε retains more geometric detail. The choice of $0.04 \times P(C)$ provides a proportional tolerance that scales with feature size, ensuring consistent simplification across polygons of different dimensions. The resulting simplified vertices are directly used as waypoints for path planning. This approach naturally adapts to arbitrary polygon geometries, generating more waypoints for complex boundaries with high curvature and fewer waypoints for simple or nearly linear shapes. Consequently, it achieves an effective balance between geometric fidelity and computational efficiency in trajectory generation. Table 4.3 summarizes the number of waypoints and their placement strategies for each shape type.

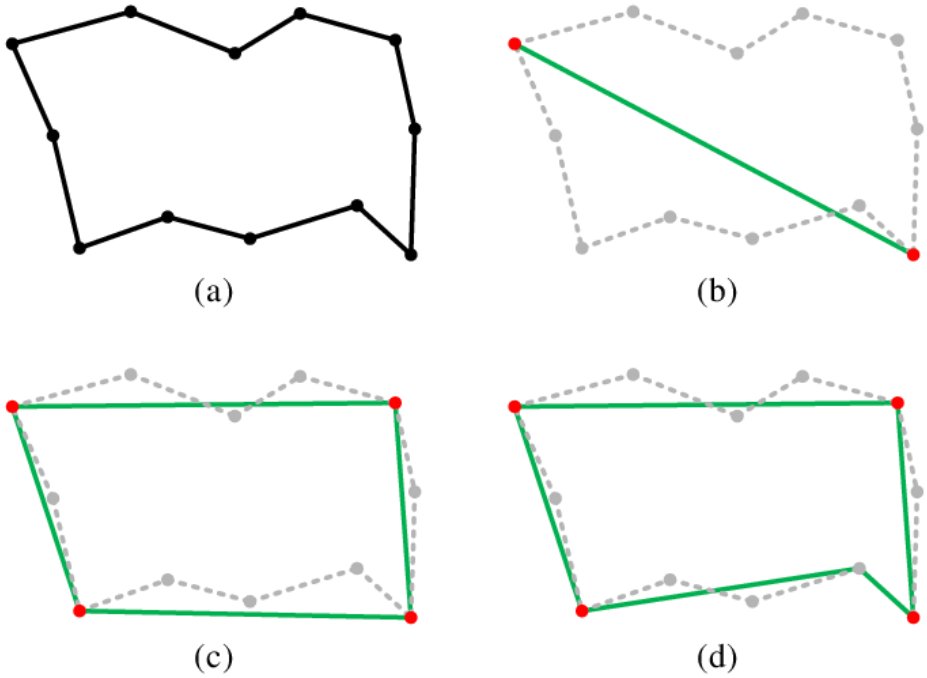


Figure 4.12: Douglas–Peucker algorithm

Table 4.3: Summary of Waypoints for Each Feature

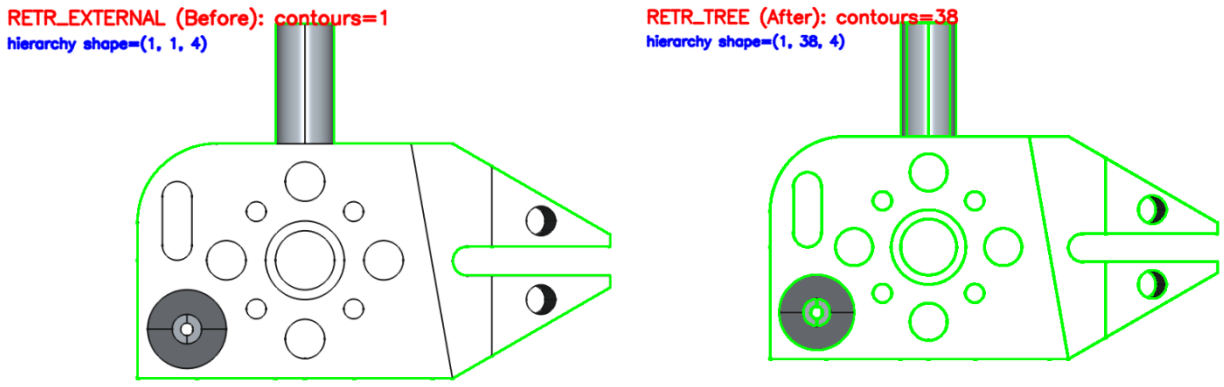
Shape Type	**Number of Waypoints**	**Placement Strategy**
Circle	3	Equally spaced at 120° intervals on the circumference
Slot (Ellipse)	6	At 45°, 90°, 135°, 225°, 270°, 315° on parametric ellipse
Rectangle	4	Opposite diagonal corners of a minimum area rectangle
Polygon	Variable	Douglas–Peucker simplified vertices ($\varepsilon = 0.04P$)
Cone	6	Same as circle, using outer contour radius

4.7 Collision Avoidance and Obstacle-Aware Path Planning

A critical requirement for any practical CMM path-planning system is the ability to avoid collisions between the probe and workspace obstacles. In CMM operations, obstacles may include fixtures, clamps, raised surfaces on the workpiece, or any geometry that the probe cannot pass through. This section presents the complete collision-avoidance framework implemented in the system, covering outer contour detection, interactive obstacle definition, obstacle-aware waypoint adjustment, and the A* graph search algorithm for shortest collision-free path generation.

4.7.1 Outer Contour Detection

The first step in collision avoidance is identifying the outer boundary of the workpiece, ensuring that all probe movements remain within the part's physical limits (Figure 4.13a). The input image is converted to binary using the same thresholding as in Section 4.2, and contours are extracted with their full hierarchy. A multi-stage filtering process is applied: contours within 10 pixels of any image edge are removed (image boundary artefacts), and contours exceeding 95% of the total image area are excluded (full frame). If no valid contours remain, a fallback strategy removes contours touching all four edges while retaining smaller candidates (Figure 4.13(b)). When feature center data is available from the detection stage, a point-in-polygon test refines the selection: the contour containing the most feature centers is selected as the workpiece outer boundary. If no contour contains any feature centers, the largest remaining contour is selected as a fallback.



(a) (b)

Figure 4.13: Contour Detection

4.7.2 Interactive Obstacle Definition

The system provides an interactive mechanism for the operator to define obstacles directly on the workpiece image through the GUI (Figure 4.14). The DrawableImageLabel widget allows the operator to draw rectangular bounding boxes by clicking and dragging on the image. Each obstacle is stored as a tuple (x_1, y_1, x_2, y_2) representing the bounding box corners in image pixel coordinates. The coordinate mapping between the widget display and the original image is handled by the get_image_coordinates method, which accounts for aspect-ratio-preserving scaling and centering.

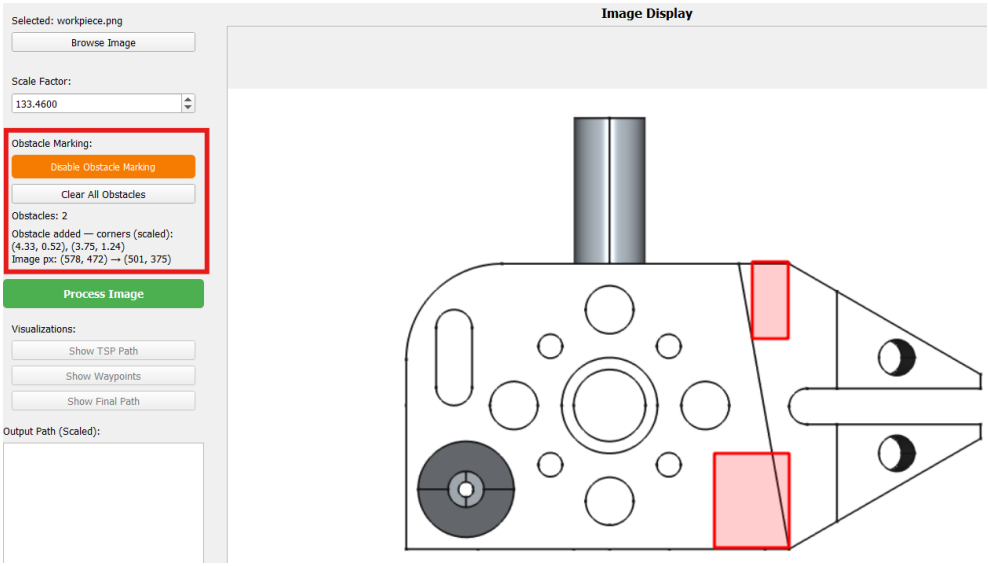


Figure 4.14: Obstacles in the Path

4.7.3 Point-in-Obstacle and Line-Obstacle Intersection Tests

The collision detection framework is based on two fundamental geometric tests (Figure 4.15). The point-in-obstacle test determines whether a given point (x, y) lies inside a rectangular obstacle defined by two opposite corners (x_1, y_1) and (x_2, y_2) as follows.

$$min(x_1, x_2) \leq x \leq max(x_1, x_2), \quad min(y_1, y_2) \leq y \leq max(y_1, y_2) \qquad (4.14)$$

The line–obstacle intersection test determines whether a path segment between two points p_1 and p_2 intersects any obstacle. For each obstacle, the algorithm first checks whether either endpoint lies inside the obstacle; if not, it evaluates intersection with each of the four edges using the counterclockwise (CCW) orientation test. Two-line segments (p_1, p_2) and (p_3, p_4) intersect if:

$$ccw(p_1, p_3, p_4) \neq ccw(p_2, p_3, p_4)) \quad \& \quad ccw(p_1, p_2, p_3) \neq ccw(p_1, p_2, p_4) \qquad (4.15)$$

Each intersection check operates in constant time O(1), and since each obstacle has four edges, the total complexity scales linearly with the number of obstacles k, resulting in O(k).

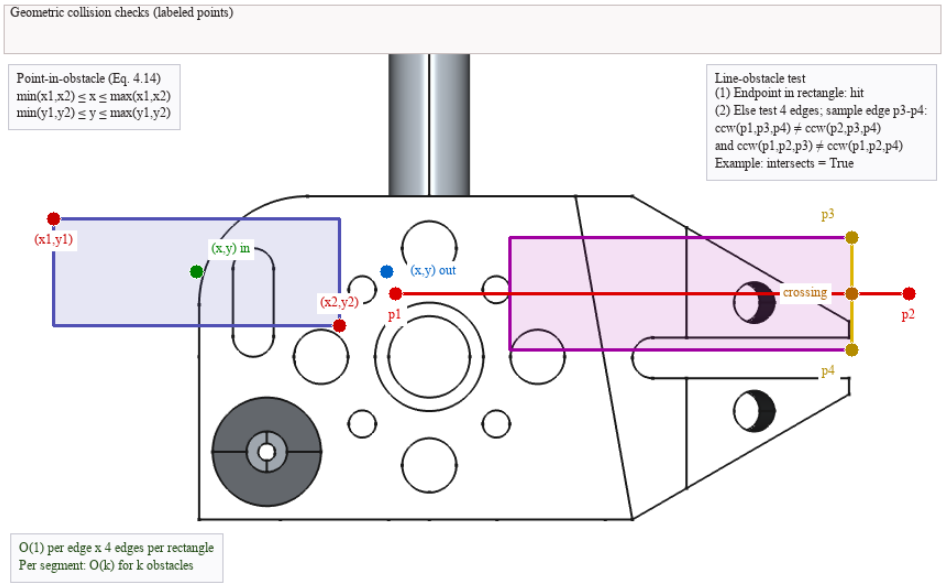


Figure 4.15: Collision Tests

4.7.4 Obstacle-Aware Waypoint Adjustment

Before computing the final path, the system first adjusts waypoints to avoid obstacles and ensure they remain within the valid workpiece boundary. This is handled by the generate_obstacle_avoiding_waypoints function, which applies three key steps: feature filtering, waypoint relocation, and contour enforcement. The algorithm first evaluates each feature against the set of known obstacles. If a feature's centroid lies within an obstacle region, or if the feature geometry overlaps with any obstacle, the feature is excluded entirely from the waypoint set. This

prevents the planner from attempting to generate trajectories through inaccessible regions. For features that pass this initial filter, each individual waypoint is examined. If a waypoint falls inside an obstacle boundary, the algorithm attempts to relocate it to the nearest valid position on the obstacle edge using the find_waypoint_on_obstacle_edge function. Should this relocation fail to produce a valid position, a fallback strategy is invoked through find_alternative_waypoint, which searches for a suitable location that lies outside all obstacles and within the outer contour. Additionally, any waypoint that falls outside the workpiece contour is similarly relocated inward using the same fallback mechanism with an interior preference. This layered approach ensures that all retained waypoints are both collision-free with respect to obstacles and geometrically valid within the machining boundary, thereby producing a feasible set of waypoints for subsequent path planning. The procedure is formalized in Algorithm 4.2.

Algorithm 4.2: Obstacle-Aware Waypoint Adjustment Algorithm

```
Input: Feature DataFrame df, Obstacles list O, Outer contour C
Output: Adjusted waypoints
For each feature f in df:
    If center of f lies inside any obstacle in O:
        Set f.waypoints = []   // remove feature
        Continue

    If shape_overlaps_obstacle(f, O):
        Set f.waypoints = []   // remove feature
        Continue
For each feature f in df:
    For each waypoint w in f.waypoints:
        If w lies inside any obstacle:
            w_new = find_waypoint_on_obstacle_edge(w, O)
            If w_new is not valid:
                w_new = find_alternative_waypoint(w, O, C)
            Replace w with w_new
        If w lies outside contour C:
            w_new = find_alternative_waypoint(w, O, C, prefer_inside=True)
            Replace w with w_new
Return df
```

The details of these three steps is as follows:

- **Feature filtering**

  If a feature's center lies inside an obstacle bounding box, the entire feature is excluded from the measurement sequence (its waypoints are set to an empty list as in Figure 4.16(c)). Additionally, the shape_overlaps_obstacle function performs a more thorough check by sampling up to 20 points along the feature's actual contour; the feature is considered blocked only if more than 50% of the sampled contour points lie within an obstacle. This conservative threshold prevents false exclusion of features that merely border an obstacle.

Algorithm 4.3: Feature–Obstacle Overlap Check

```
Input: Feature f, Obstacles list O
Output: Boolean (True = blocked)
Sample up to 20 points from boundary of feature f
count_inside = 0
For each sampled point p:
      If p lies inside any obstacle in O:
          count_inside += 1
      If (count_inside / total_samples) > 0.5:
          Return True
Else:
      Return False
```

- **Waypoint relocation**

  For features not blocked by obstacles, each waypoint is checked individually. If a waypoint lies inside an obstacle, the system attempts to find the nearest valid alternative using a two-stage search. First, the waypoint is projected to the closest point on the obstacle boundary, offset by a 5-pixel margin. If this fails, a spiral search tests candidate points at increasing radii (10 to 100 pixels, in 10-pixel increments as in Figure 4.16(d)) and angles (0° to 360° in 15° increments). Each candidate is validated to ensure it lies outside all obstacles and within the outer contour.

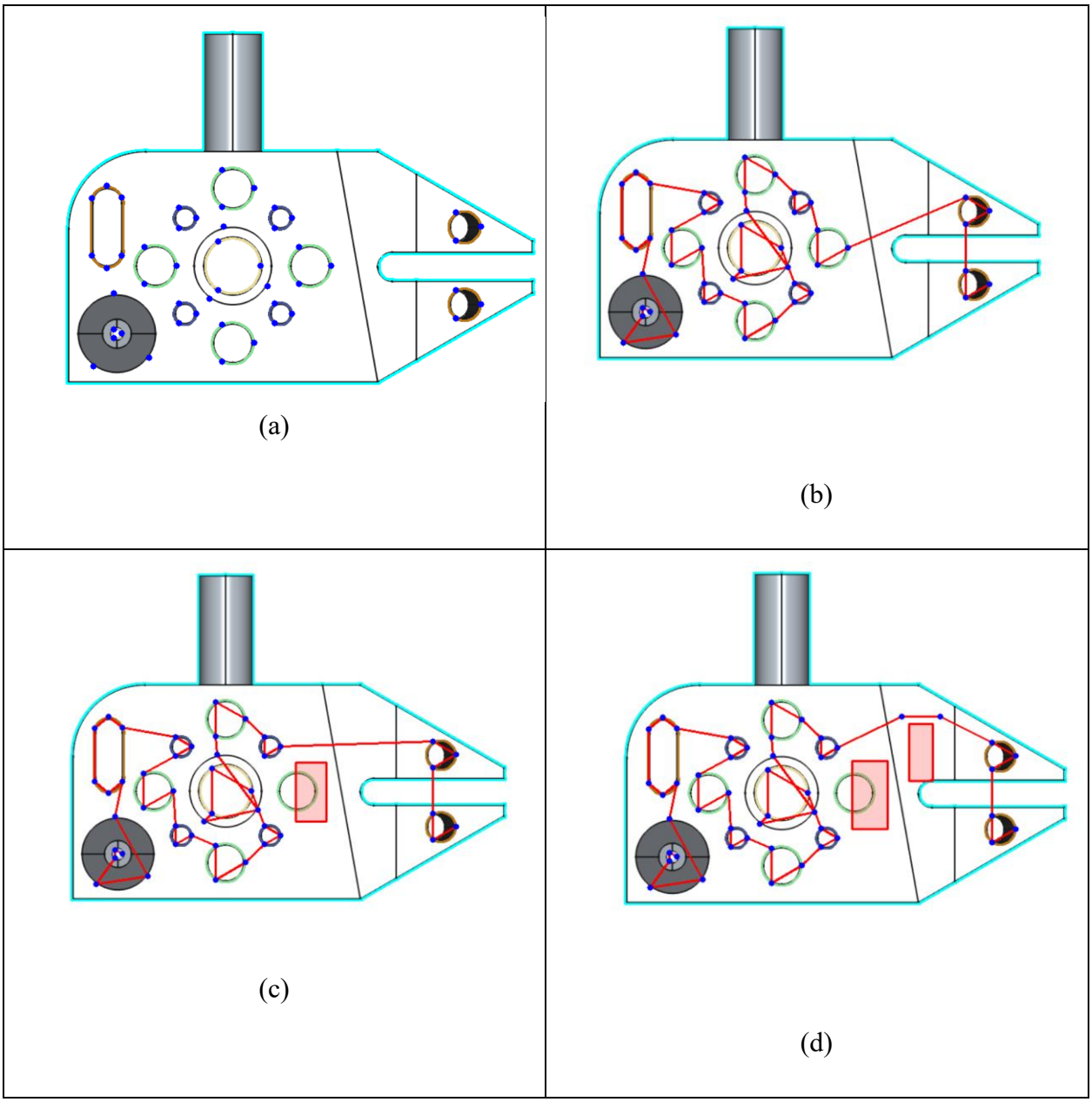


Figure 4.16: Illustration of obstacle-aware path planning (a) Waypoints on the features, (b) Path connecting all waypoints, (c) Obstacle covering feature, (d) Obstacle blocking the path

Algorithm 4.4: Waypoint Projection to Obstacle Edge

```
Input: Waypoint w(x, y), Obstacles list O
Output: Adjusted waypoint w_new
For each obstacle defined by (x1, y1, x2, y2):
    Compute closest point on rectangle boundary:
        x_new = clamp(x, min(x1,x2), max(x1,x2))
        y_new = clamp(y, min(y1,y2), max(y1,y2))
```

```
    Move point slightly outward (offset = 5 pixels)
    If (x_new, y_new) is outside obstacle:
        Return (x_new, y_new)
Return None   // if no valid projection found
```

Algorithm 4.5: Spiral Search for Alternative Waypoint Algorithm

```
Input: Waypoint w(x, y), Obstacles list O, Contour C
Output: Valid waypoint w_new
For radius r = 10 to 100 step 10:
    For angle θ = 0° to 360° step 15°:
        x_new = x + r * cos(θ)
        y_new = y + r * sin(θ)
        If point is outside all obstacles AND inside contour C:
            Return (x_new, y_new)
Return original w   // fallback if no valid point found
```

- **Contour boundary enforcement**

  Waypoints that lie outside the workpiece's outer contour are similarly relocated using the spiral search, with candidates restricted to within the contour boundary.

Algorithm 4.6: Contour Boundary Check

```
Input: Point p, Contour C
Output: Boolean
Use point-in-polygon test (e.g., cv2.pointPolygonTest)
If p is inside contour C:
    Return True
Else:
    Return False
```

4.8 A* Graph Search for Collision-Free Pathfinding

A*, introduced in Section 2.2.2, is selected for local collision-free routing because of its guaranteed optimality with an admissible heuristic. The algorithm finds the shortest safe path between two points by exploring promising routes first, using a distance estimate to the goal to focus its search.

Unlike conventional grid-based approaches that discretize the entire workspace, this implementation operates in continuous space by constructing a visibility graph from key geometric points, thereby significantly reducing computational complexity while maintaining accurate obstacle avoidance. The following describes the graph construction process and the subsequent A* search used for collision-free path planning.

- **Graph construction**

For each pair of consecutive waypoints, the algorithm first evaluates whether a direct straight-line connection is collision-free. If there is no intersection with obstacles, the direct path is accepted. Otherwise, a visibility graph is constructed to determine an alternative route. The graph consists of the start point, goal point, and corner points of all obstacles, each slightly offset outward to ensure a safety margin. Any candidate points that lie inside obstacles or outside the workpiece boundary are excluded. Edges are then created between pairs of nodes only if the connecting line segment does not intersect any obstacle and both endpoints lie within the valid contour. Each edge is assigned a weight equal to the Euclidean distance between the two nodes, allowing the problem to be formulated as a shortest-path search on a weighted graph.

- **A* search with Euclidean heuristic**

Once this graph is formed, A* is applied to determine the shortest feasible route. The algorithm maintains a priority queue (min-heap) of candidate nodes ordered by the cost function:

$$f(w) = g(w) + h(w) \qquad (4.16)$$

where g(w) is the actual travel cost from the start to the current node, and h(w) is the Euclidean distance from the current node to the goal. Since the Euclidean distance never overestimates the remaining cost, it is an admissible heuristic and guarantees the optimality of the search. The complete step-by-step execution of this algorithm, as implemented in the path planning module, is illustrated in 4.17.

At each iteration, the algorithm initializes two lists: the open list, seeded with the start node at g = 0, and an empty closed list. The node with the lowest f-value is then selected from the open list

and moved to the closed list, marking it as evaluated. As shown in the decision branch at the top of Figure 4.17, if the open list becomes empty before the goal is reached, the algorithm terminates and returns a failure state, indicating that no collision-free path exists between the start and goal waypoints.

For the selected node, all neighbouring walkable nodes are expanded, subject to clearance and collision constraints derived from the PNG workspace grid. For each neighbor, the algorithm computes a tentative g-score; if this improves on the previously known best path to that neighbor, the neighbor's predecessor and g-score are updated. This update logic corresponds to the lower decision diamond in Figure 4.17, where a neighbour is added or refreshed in the open list if g_new < g(n). The search terminates when the goal node is expanded, at which point the optimal path is reconstructed by backtracking through predecessor pointers. This backtracking step is shown as the "Reconstruct path" output block on the right branch of Figure 4.17.

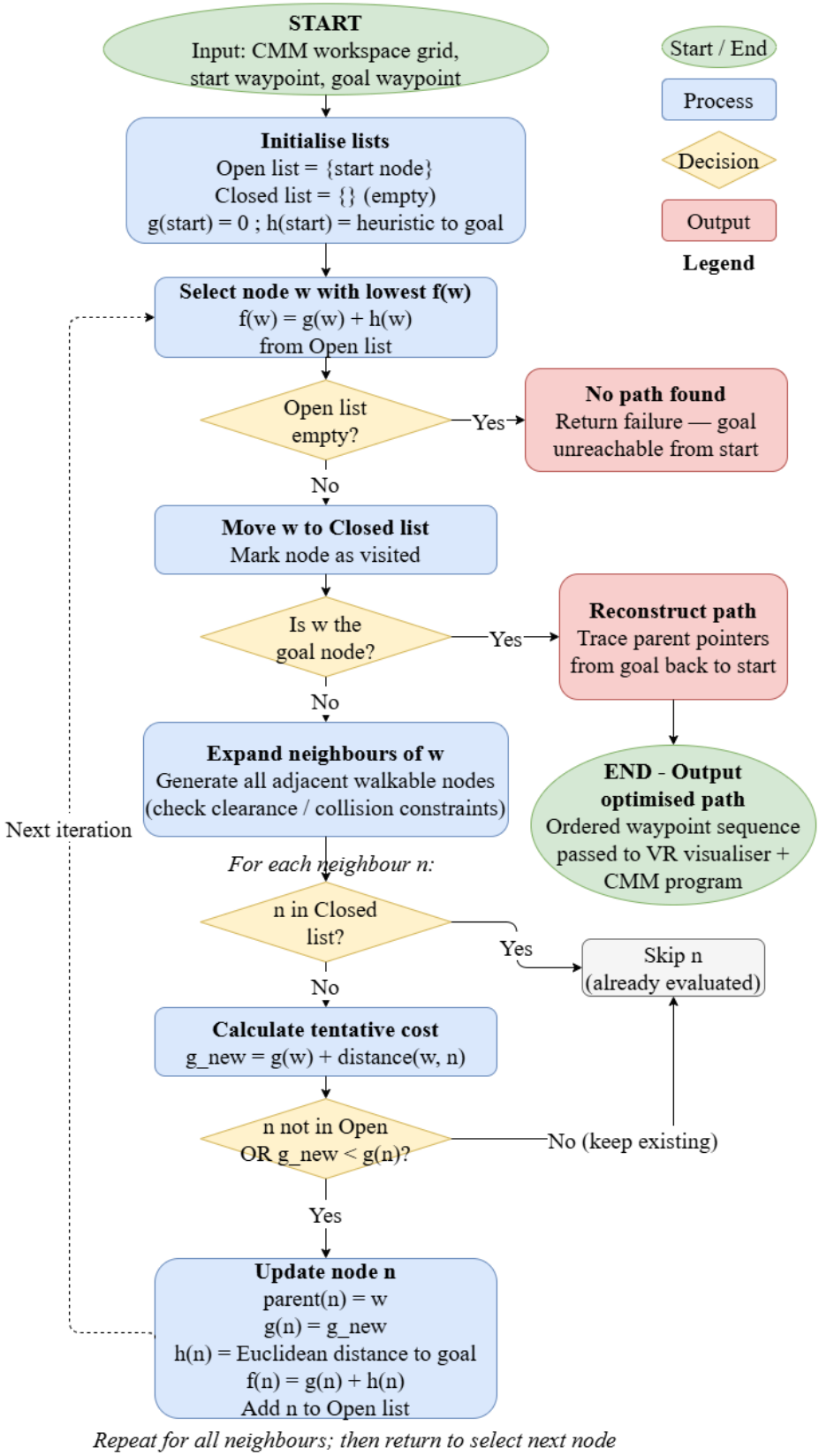


Figure 4.17: Flowchart of the A search algorithm in the CMM path planning module*

Figures 4.18 and 4.19 illustrate the working principle of A*, where **S** denotes the start point and **G** denotes the goal point, while the black cells represent obstacles that cannot be traversed. Each free cell shows the cost values used by the algorithm, typically written as $g + h$, where g is the accumulated cost from the start and h is the estimated distance to the goal. Conceptually, the figure explains A* using a grid representation because it makes the search process easier to visualize. However, in the implemented path-planning module, collision is checked geometrically using straight-line intersection tests with rectangular obstacles and outer contour validation, rather than by expanding every cell of a dense workspace grid. Thus, the figure should be understood as an illustration of A* search logic, whereas the actual implementation applies the same principle on a visibility graph in a continuous space.

The A* formulation guarantees that the returned path is the shortest collision-free route through the visibility graph, provided the heuristic is admissible and consistent. The Euclidean distance heuristic satisfies both properties in the plane, ensuring optimality. The time complexity is $O(V^2 \log V)$ in the worst case, where V is the number of nodes in the visibility graph, which is typically small (4 nodes per obstacle plus start and goal).

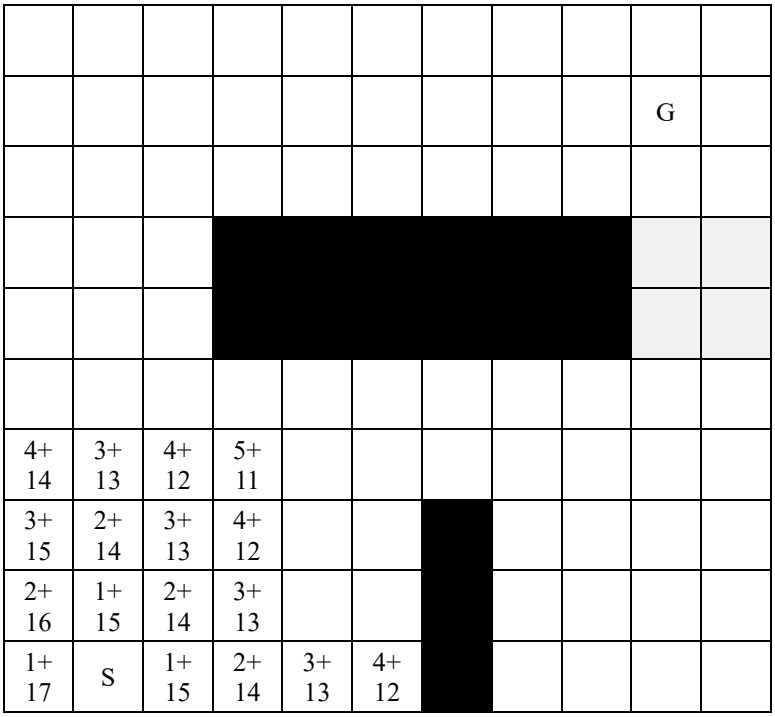


Figure 4.18: Grid Representation of A Algorithm*

	9+9	10+8	11+7	12+6	13+5	14+4	15+3			
9+9	8+8	9+7	10+6	11+5	12+4	13+3	14+2	15+1	G	
8+10	7+9	8+8	9+7	10+6	11+5	12+4	13+3	14+2	15+1	
7+11	6+10	7+9							14+2	15+3
6+12	5+11	6+10							13+3	14+4
5+13	4+12	5+11	6+10	7+9	8+8	9+7	10+6	11+5	12+4	13+5
4+14	3+13	4+12	5+11	6+10	7+9	8+8	10+7	11+6	12+5	
3+15	2+14	3+13	4+12	5+11	6+10					
2+16	1+15	2+14	3+13	4+12	5+11					
1+17	S	1+15	2+14	3+13	4+12					

*Figure 4.19: Final Path Using A**

Algorithm 4.7: A Collision-Free Pathfinding on Visibility Graph*

```
Input: start, goal, obstacles[], outer_contour
Output: Shortest collision-free path as list of points
IF direct line start->goal is obstacle-free:
  RETURN [start, goal]
// Build visibility graph
nodes = [start] + obstacle_corner_points(margin=10) + [goal]
Filter nodes: remove those in obstacles or outside contour
FOR each pair (i, j) in nodes:
  IF line(nodes[i], nodes[j]) does not intersect any obstacle
     AND both nodes within outer_contour:
    Add edge(i, j) with weight = euclidean_distance
// A* search
open_set = min-heap with (h(start), start_idx)
g_score[start] = 0; closed = {}
WHILE open_set not empty:
  current = pop node with min f = g + h
  IF current == goal: reconstruct and RETURN path
  closed.add(current)
```

```
  FOR each neighbor of current:
    IF neighbor in closed: SKIP
    tentative_g = g[current] + edge_weight
    IF tentative_g < g[neighbor]:
      predecessor[neighbor] = current
      g[neighbor] = tentative_g
      push (tentative_g + h(neighbor), neighbor) to open_set
RETURN fallback_path or None (no path found)
```

4.8.1 Waypoint Traversal Ordering

For each feature, the measurement waypoints must be visited in an order to minimize transition distance from the previous feature to the next feature. The optimize_feature_waypoints function implements a bidirectional evaluation strategy. Given the current probe position (exit point of the previous feature) and the waypoints of the current feature, the algorithm identifies the entry waypoint closest to the current position. It then evaluates two possible traversal directions, forward and backward, from the entry point. If information about the next feature's waypoints is available, the direction whose exit point is closest to any waypoint of the next feature is selected. Otherwise, the direction with the shorter total internal travel distance is chosen.

4.8.2 Full Path Assembly

The complete measurement path is assembled by the generate_obstacle_aware_path function, which processes features in TSP order and concatenates the inter-feature and intra-feature movements. For each feature, the function first reorders the feature's waypoints using the traversal ordering algorithm. It then checks, for each consecutive waypoint, whether a direct connection is obstacle-free. If the direct line intersects an obstacle, the A* path-finding algorithm is invoked to find a collision-free detour, and the resulting intermediate waypoints are inserted into the path. After processing all features, consecutive duplicate points are removed to produce a clean path.

4.8.3 Coordinate Transformation and Final Output

All computations up to this point (feature detection, classification, TSP sequencing, and A* path planning) are performed in the image pixel coordinate system, where the origin is at the top-left corner with the y-axis pointing downward. The CMM, however, operates in a Cartesian coordinate

system with the origin typically at the bottom-left and the y-axis pointing upward, measured in physical units. Bridging these two domains requires two operations: a y-coordinate flip and a scale conversion from pixels to inches. The scale factor is established by relating a known physical dimension of the workpiece to its corresponding pixel span in the image. If the workpiece has a known physical width of W inches and occupies P pixels in the image, the scale factor is:

$$s = P / W \text{ (pixels per inch)} \quad (4.17)$$

In this work, a one-time calibration procedure yields a default scale factor of 133.46 pixels per inch. To convert any pixel measurement to inches, both coordinates are divided by s. This conversion assumes that the image is an undistorted, to-scale representation of the part with a square pixel aspect ratio (1:1), so that a single scale factor applies in both the x- and y-directions.

In summary, the pixel values reported throughout this chapter originate directly from the 864 × 542 image grid as the native output of the computer vision algorithms applied during feature detection (Section 4.3) and classification (Section 4.4). The coordinate transformation, combining the y-axis flip with the pixel-to-inch scaling, bridges these image-domain measurements to the physical dimensions required by the CMM, ensuring that the final measurement coordinates presented in Table 5.4 correspond to real-world positions on the workpiece.

4.9 Summary and Discussion

This chapter presented a comprehensive, modular path planning system for CMM inspection that integrates computer vision, combinatorial optimization, and collision-aware pathfinding into a single interactive application. The system takes a workpiece image as input and produces an optimized, collision-free measurement path with minimal operator intervention. The final interface of the path planning application is shown in Figure 4.20.

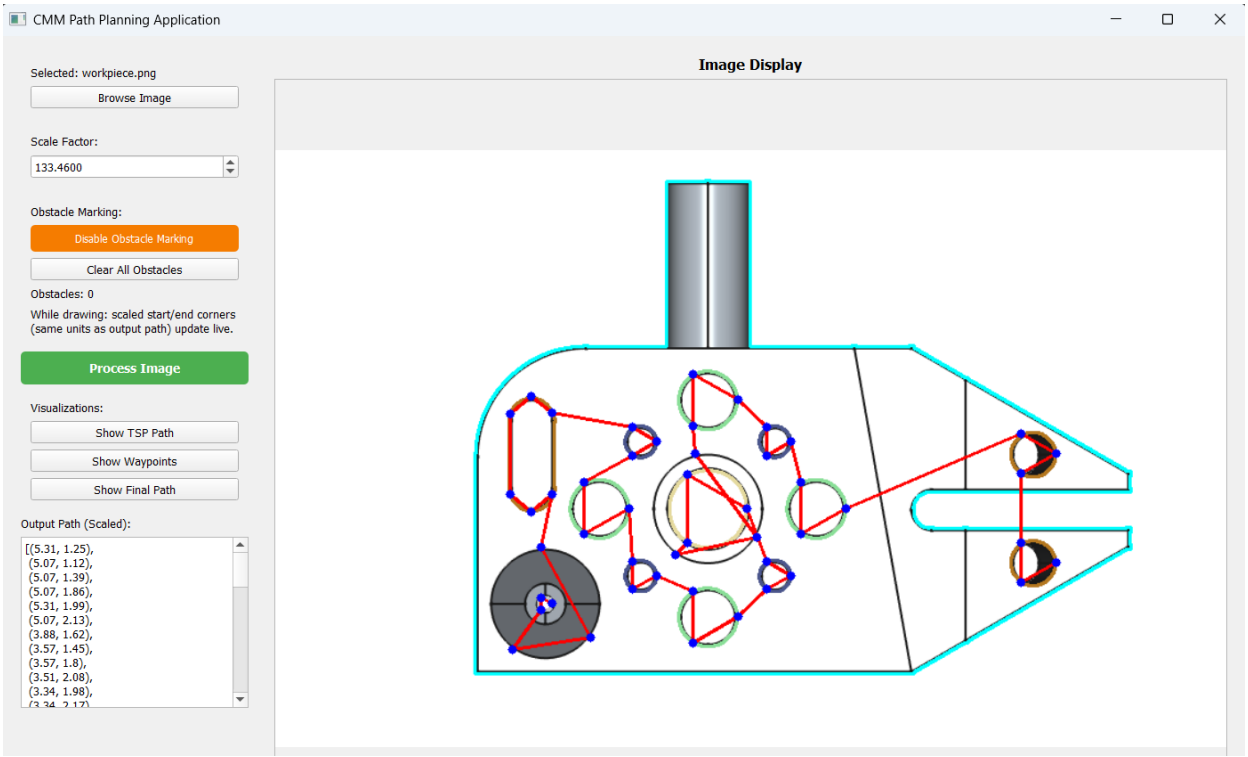


Figure 4.20: Path Planning App Interface

The key contributions and capabilities demonstrated in this chapter are as follows.

The shape detection and classification pipeline, built on OpenCV's contour detection, polygon approximation and circularity analysis, automatically identifies and categorizes geometric features from the workpiece image. The TSP-based sequence optimization, using a nearest-neighbor heuristic refined by 2-opt local search, produces efficient visitation orders that minimize total probe travel distance. Shape-specific waypoint generation ensures that each feature type receives an appropriate set of measurement contact points. The collision avoidance framework, comprising outer contour detection, interactive obstacle definition, obstacle-aware waypoint adjustment, and A* visibility-graph pathfinding, ensures that the generated path avoids all defined obstacles while remaining within the workpiece boundary. The coordinate transformation stage converts the planned path from image coordinates to CMM-executable physical coordinates.

Chapter 5: Results, Case Study, and Evaluation

5.1 Introduction

Building on the path planning system described in Chapter 4 and the VR training environment developed in Chapter 3, this chapter presents a comprehensive evaluation of the VR-based CMM training and inspection planning framework developed in this thesis. The evaluation is structured around three complementary dimensions: (1) metrological fidelity and measurement accuracy; (2) path planning performance demonstrated through a practical case study on a representative workpiece; and (3) user experience and usability assessed through a standardized questionnaire administered to 30 participants.

Section 5.2 describes the experimental configurations and hardware. Sections 5.3–5.5 evaluate measurement fidelity, including feature-wise accuracy, agreement analysis, and correlation between VR and physical CMM measurements. Sections 5.6–5.8 present the case study applying the path planning system to a 13-feature workpiece under two scenarios: obstacle-free and with a defined obstacle. Section 5.9 presents the comparative analysis of both cases against the manual planning baseline. Section 5.10 evaluates path planning performance and scalability against baseline methods. Section 5.11 presents the results of the usability questionnaire. A chapter summary concludes the evaluation.

5.2 Experimental Configurations

To support traceability and reproducibility, the experimental setup is documented as follows:

- **Hardware (PC):** CPU: Intel Core i7 10th Generation; GPU: NVIDIA GTX 1660Ti; RAM: 16 GB; OS: Windows 10
- **Hardware (VR):** Meta Quest 2 / Oculus Quest 2 with hand controllers
- **Software environment:** Unity 2022.3.20f1; XR Plug-in: OpenXR; XR Interaction Toolkit; Physics fixed timestep: 0.02 s; Frame rate cap: 60 FPS

- **CMM simulation model:** Bridge-type virtual CMM, 3-axis Cartesian motion; Axis limits: X = 16 in, Y = 20 in, Z = 14 in

- **Physical CMM specifications:** Resolution of 0.000020 inches (0.5 μm); repeatability of 0.00013 inches (3.3 μm); linear accuracy of 0.00013 inches + 0.000005 inches per inch (3.3 μm + workpiece length/200). These parameters ensure reliable, repeatable dimensional inspection and serve as the benchmark for VR measurement accuracy evaluated in Section 5.3.

- **Evaluation workpiece:** Blue anodized aluminum training widget includes prismatic features (planes, steps, circles, cones, slots, and line features) in all orientations, consistent with industrial CMM training curricula (see Figure 5.1 context in Section 5.3)

5.3 Measurement Fidelity and Accuracy Results

5.3.1 Definition of Measurement Accuracy Metrics

To evaluate measurement fidelity, each measured dimension was compared against the nominal reference value from the CAD file. Because the VR environment geometry is built directly from CAD, the CAD nominal dimensions serve as the ideal references. Let D_{ref} denote the reference dimension, D_{VR} the dimension measured in the VR system, and D_{CMM} the dimension measured on the physical CMM. The signed errors are:

$$\left[e_{VR} = D_{VR} - D_{ref}, \qquad e_{CMM} = D_{CMM} - D_{ref}\right] \tag{5.1}$$

Key accuracy metrics reported include:

- **Absolute error**

$$|e| = |D - D_{ref}| \tag{5.2}$$

- **Mean error (bias) across N measurements:**

$$\bar{e} = (1/N)\, \Sigma\, e_i \tag{5.3}$$

- **Standard deviation (precision/repeatability)**

$$\sigma = \sqrt{[(1/(N-1))\, \Sigma(e_i - \bar{e})^2]} \tag{5.4}$$

- **Root mean square error (RMSE)**

$$RMSE = \sqrt{[(1/N)\ \Sigma\ e_i^2]} \qquad (5.5)$$

These metrics together quantify both systematic deviation (bias) and variability (repeatability), which are critical in coordinate metrology contexts.

5.3.2 Feature-Wise Accuracy Results

Measurement results are reported by feature class, as different geometric features require distinct probing strategies and point-acquisition methods. For point measurements, the primary metric is coordinate capture error. For line measurements, accuracy is assessed through point-to-point distance errors. Plane measurements report deviations in computed plane parameters and derived distances. Circular features use a three-point probing strategy, with diameter/radius compared to nominal. Complex features such as cones and slots are evaluated by feature-specific attributes (angle, width, depth).

Table 5.1 presents the structured feature-wise accuracy results, and Figure 5.1 displays the corresponding RMSE bar chart. Figure 5.2 shows boxplots of absolute errors across feature types. To contextualize these results for training purposes: typical CMM positional tolerances in industrial inspection are on the order of 0.001-0.01 inches for linear features and 0.05°-0.10° for angular features. The VR system's RMSE values for linear features (0.0009-0.0016 inches) fall within this range, meaning trainees develop measurement intuition at accuracy levels representative of real-machine conditions. The cone angle RMSE of 0.21° is slightly above the typical angular tolerance threshold, which is expected given that angular features are more sensitive to the discretization effects inherent in real-time physics simulation. This does not undermine the system's training value for angular features, trainees still learn the correct probing strategy and procedure, but it does suggest that cone angle measurements should be interpreted as approximate in the VR environment. Overall, the results confirm that the VR system is sufficiently accurate to serve as a credible training proxy for physical CMM operation across all tested feature types.

Table 5.1: Feature Measurement Accuracy Comparison (VR vs. Physical CMM)

Feature	**Dimension**	D_{ref} *(in)*	D_{VR} *mean±SD*	D_{CMM} *mean±SD*	e_{VR}	e_{CMM}	$RMSE_{VR}$	$RMSE_{CMM}$
Circle	Diameter	1.000	1.0007 ±0.0005	1.0002 ±0.0002	0.0007	0.0002	0.0009	0.0003
Plane	Flatness/dist.	2.000	2.0011 ±0.0006	2.0003 ±0.0002	0.0011	0.0003	0.0014	0.0004
Line	Length	4.000	3.9988 ±0.0008	3.9996 ±0.0003	0.0012	0.0004	0.0016	0.0005
Slot	Width	0.375	0.3762 ±0.0007	0.3753 ±0.0002	0.0012	0.0003	0.0015	0.0004
Cone	Angle	60.000°	60.18 ±0.09	60.04 ±0.03	0.18°	0.04°	0.21°	0.05°

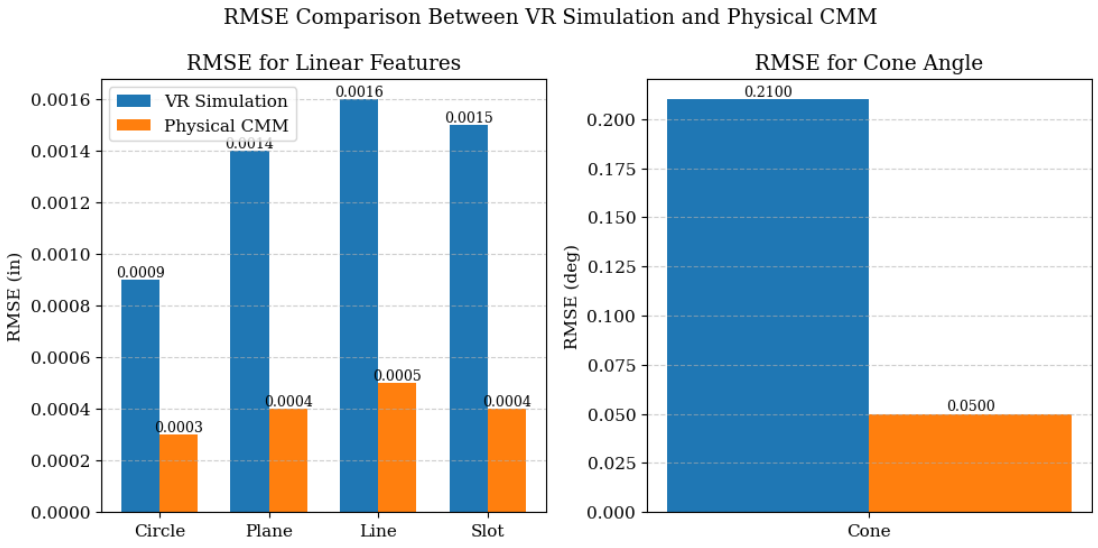


Figure 5.1: Bar chart of RMSE by feature type (VR vs. physical CMM)

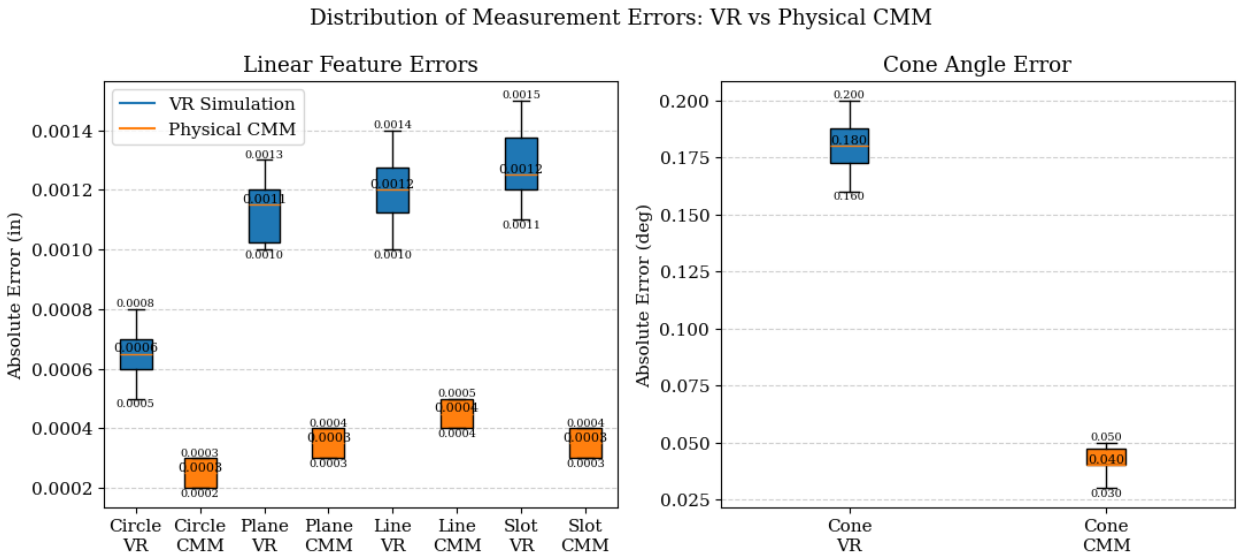


Figure 5.2: Boxplots of absolute errors for VR and physical CMM across feature types

5.4 Agreement Between VR and Physical CMM Measurements

In addition to accuracy relative to reference dimensions, it is essential to evaluate the agreement between VR measurements and physical CMM measurements directly, since the primary purpose of the VR system is to serve as a training proxy for real CMM operation. Strong agreement supports the validity of the simulation platform. Let the difference between the two measurement systems for the i-th observation be:

$$d_i = Dv^{R}{}_{,i} - Dc^{MM}{}_{,i} \qquad (5.6)$$

The mean difference, standard deviation of differences, and limits of agreement (LOA) are computed as:

$$\bar{\bar{a}} = (1/N)\ \Sigma\ d_i \qquad (5.7)$$

$$s_e = \sqrt{[(1/(N-1))\ \Sigma(d_i - \bar{\bar{a}})^2]} \qquad (5.8)$$

$$LOA = \bar{\bar{a}} \pm 1.96\ s_e \qquad (5.9)$$

The limits of agreement represent the interval within which approximately 95% of all measurement differences between the VR and physical CMM systems are expected to lie.

5.4.1 Sample Measurement Data

Representative results from 20 sample readings are shown in Table 5.2, and the computed agreement statistics are shown in Table 5.3.

Table 5.2: Comparison of VR and Physical CMM Measurements

Feature	Reference (in)	Mean VR Measurement	Mean CMM Measurement	Difference d_i
Circle diameter	1.0000	1.0008	1.0002	0.0006
Plane distance	2.0000	2.0011	2.0003	0.0008
Line length	4.0000	3.9989	3.9996	−0.0007
Slot width	0.3750	0.3762	0.3753	0.0009

Table 5.3: Agreement Statistics Between VR and Physical CMM Measurements

Statistic	Value
Mean difference ($\bar{a}$)	0.0004 in
Standard deviation (s_e)	0.0007 in
Lower limit of agreement	−0.0010 in
Upper limit of agreement	0.0018 in

These results confirm that VR-based measurements closely match physical CMM measurements, with all differences falling within a narrow tolerance range acceptable for training purposes.

5.5 Bland–Altman Agreement Analysis and Correlation

A Bland–Altman plot was used to visualize the agreement between VR-based and physical CMM measurements. This method plots the difference between the two measurement systems against

their mean, allowing systematic bias and dispersion to be observed. As shown in Figure 5.3, measurement differences lie within the limits of agreement, demonstrating strong consistency between the two systems.

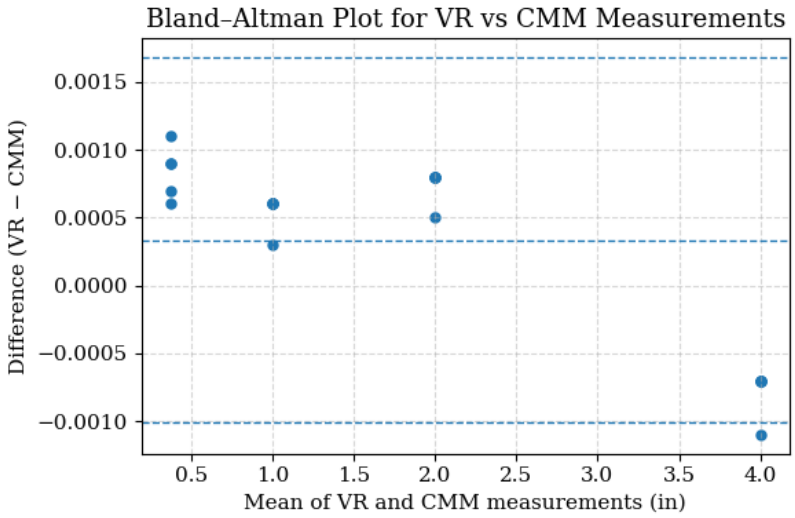


Figure 5.3: Bland–Altman plot

5.5.1 Correlation Between VR and Physical CMM Measurements

A scatter plot and the coefficient of determination R^2 were computed to quantify the linear relationship between VR and physical CMM measurements:

$$R^2 = \frac{[\sum(x_i-\bar{x})(y_i-\bar{y})]^2}{\sum(x_i-\bar{x})^2 \cdot \sum(y_i-\bar{y})^2} \qquad (5.10)$$

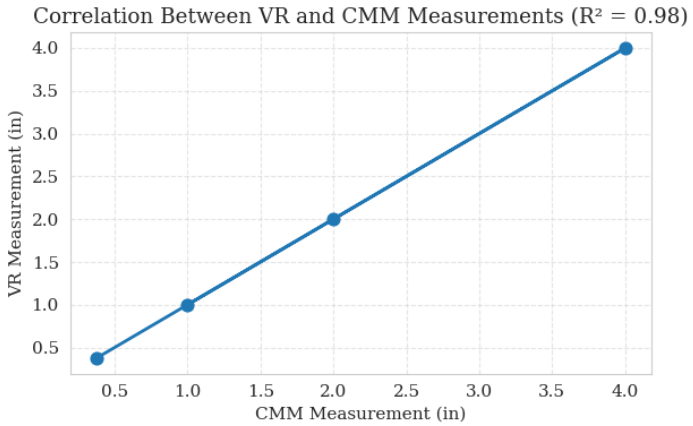


Figure 5.4: Scatter plot of VR

5.6 Case Study

5.6.1 Introduction

This section presents a case study that applies the path-planning system described in Chapter 4 to demonstrate the system from automatic feature detection and collision-free path generation to evaluate its performance under two distinct scenarios: one without workspace obstacles, and one with a user-defined obstacle simulating a fixture or clamp. The section concludes a quantitative comparison between the automated path plan and a manual plan. The case study focuses entirely on the application, results, and analysis.

5.6.2 Workpiece Description

The workpiece for this case study is a mechanical part containing a diverse set of geometric features representative of typical CMM inspection tasks. The part is 6 inches wide and 4.5 inches tall in the inches coordinate system (corresponding to the pixel extent divided by a calibration factor of 133.46 pixels/inch).

The following features are present:

- **Circles (10):** Varying radii from approximately 14 px in (small alignment holes) to 35 px in, including a ring of six small circles surrounding the central bore.
- **Central hole (1):** Large through-hole at the geometric centre, radius ≈51 px, representing a primary datum bore critical for dimensional inspection.
- **Slot (1):** Elongated elliptical feature on the left side, major axis at an angle, representative of fastener clearance slots.
- **Cone (1):** Concentric circle pair in the lower-left region classified as a cone (countersunk hole or chamfered feature) based on the radius ratio between inner and outer circles.

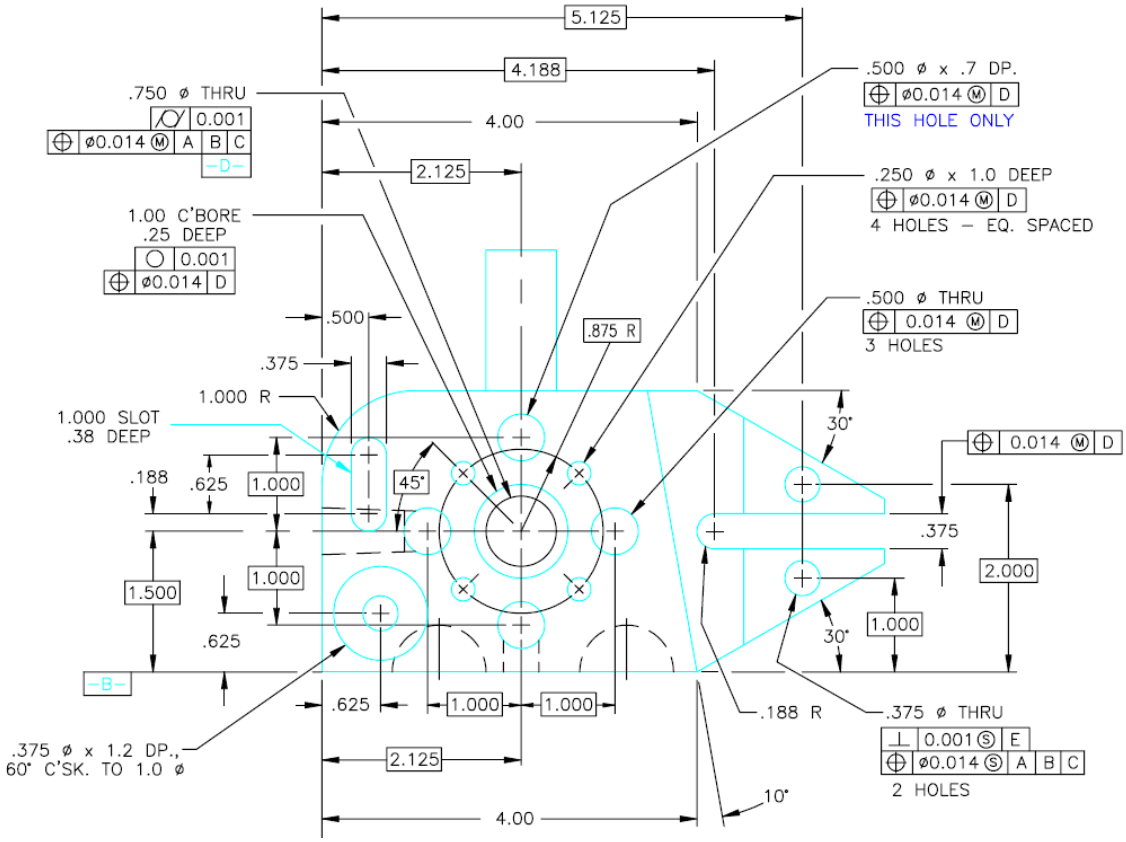


Figure 5.5: CAD Drawing of Workpiece

The measurement identified all 14 features. Table 5.4 lists all detected features, including their types, areas, radii, and centroid coordinates, used as input to the TSP sequence optimizer and waypoint generation modules.

The system correctly identified all 14 features, including the concentric circle pair classified as a cone (Feature 13), the large central hole (Feature 5), and the elongated slot (Feature 12). The calibration scale factor of 133.46 px/unit was applied throughout to convert all coordinates from image pixel space to the CMM's physical coordinate frame.

Table 5.4: Detected Features on the Sample Workpiece

Feature #	Shape	Centre X (px)	Centre Y (px)	Centre X (in)	Centre Y (in)
0	Circle	688.13	374.56	5.1247	0.9997
1	Circle	688.19	275.98	5.1252	2.0003
2	Circle	491.16	325.27	3.1252	1.5000
3	Circle	453.56	264.42	2.7435	2.1177
4	Circle	392.66	226.79	2.1253	2.4997
5	Hole	392.61	325.27	2.1248	1.5000
7	Circle	453.53	386.30	2.7432	0.8804
8	Circle	392.67	423.81	2.1254	0.4997
9	Circle	331.66	386.34	1.5061	0.8801
10	Circle	294.14	325.28	1.1252	1.4999
11	Circle	331.61	264.38	1.5055	2.1181
12	Slot	232.43	275.93	0.4987	2.0009
13	Cone	244.87	411.46	0.6251	0.6250

5.7 Case 1: Path Planning without Obstacles

5.7.1 Scenario Description

In Case 1, the path planning was executed on the workpiece with no obstacles defined in the workspace. This represents an unobstructed inspection scenario where the CMM probe has free access to all 14 features, with no fixtures, clamps, or other geometry interfering with its probe trajectory.

5.7.2 TSP Sequence Optimization

The TSP solver (nearest-neighbour heuristic followed by 2-opt local improvement) produced an optimized visitation sequence for all 14 features. The algorithm grouped the two right-side circles (Features 0–1) together, visited the central cluster (Features 3–12) in a sweep pattern, and concluded with the left-side features (Slot 12 and Cone 13). The resulting tour eliminated crossing paths and grouped spatially proximate features for consecutive measurement, as confirmed by visual inspection of the TSP path (Figure 5.6).

5.7.3 Final Path and Output Coordinates

The complete probe path, including all shape-specific measurement waypoints and inter-feature transit segments, was generated and visualized. The final path consists of 48 waypoints: three contact points per circle, six contact points for the slot, and two diagonal corner points for rectangular features, with smooth transitions between consecutive features. No collision-avoidance detours were required.

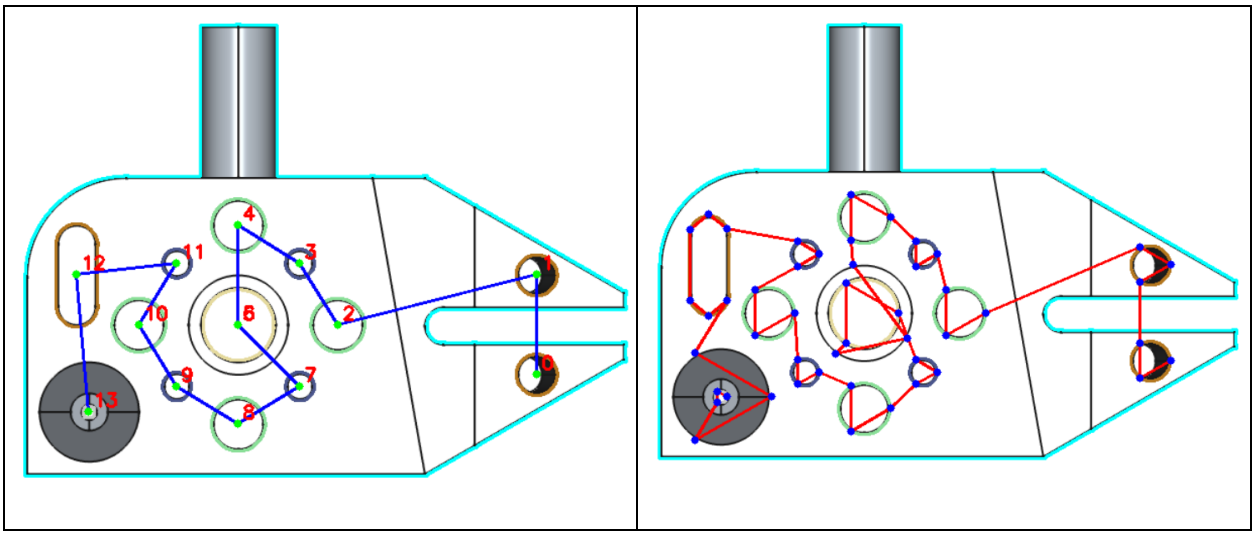


Figure 5.6: TSP-optimised visitation sequence and Final Path for Case 1

The total path length for Case 1 was 18.90 inches. Table 5.5 presents selected output waypoints, and Table 5.6 summarises the key metrics.

Table 5.5: Output Path Coordinates for Case 1

Feature #	WP #	X (px)	Y (px)	X (in)	Y (in)
0	0	709.00	374.00	5.3125	1.2513
0	1	677.00	392.00	5.0727	1.1164
0	2	677.00	356.00	5.0727	1.3862
1	0	709.00	275.00	5.3125	1.9931
1	1	677.00	293.00	5.0727	1.8582
1	2	677.00	257.00	5.0727	2.1280
2	0	518.00	325.00	3.8813	1.6185
2	1	477.00	348.00	3.5741	1.4461
2	2	477.00	301.00	3.5741	1.7983
3	0	468.00	264.00	3.5067	2.0755
3	1	446.00	277.00	3.3418	1.9781
3	2	446.00	251.00	3.3418	2.1729
4	0	420.00	226.00	3.1470	2.3603

4	1	379.00	250.00	2.8398	2.1804
4	2	379.00	203.00	2.8398	2.5326
5	0	381.00	275.00	2.8548	1.9931
5	1	437.00	351.00	3.2744	1.4236
5	2	363.00	367.00	2.7199	1.3038
6	0	428.00	325.00	3.2070	1.6185
6	1	374.00	356.00	2.8023	1.3862
6	2	374.00	294.00	2.8023	1.8507
7	0	468.00	386.00	3.5067	1.1614
7	1	446.00	398.00	3.3418	1.0715
7	2	446.00	373.00	3.3418	1.2588
8	0	420.00	423.00	3.1470	0.8842
8	1	379.00	447.00	2.8398	0.7043
8	2	379.00	400.00	2.8398	1.0565
9	0	346.00	386.00	2.5925	1.1614
9	1	324.00	398.00	2.4277	1.0715
9	2	324.00	373.00	2.4277	1.2588
10	0	321.00	325.00	2.4052	1.6185
10	1	280.00	348.00	2.0980	1.4461
10	2	280.00	301.00	2.0980	1.7983
11	0	346.00	264.00	2.5925	2.0755
11	1	324.00	277.00	2.4277	1.9781
11	2	324.00	251.00	2.4277	2.1729
12	0	213.07	239.15	1.5965	2.2617
12	1	232.22	223.81	1.7400	2.3767
12	2	251.50	238.99	1.8844	2.2629
12	3	251.79	312.71	1.8866	1.7106
12	4	232.64	328.05	1.7431	1.5956

12	5	213.36	312.86	1.5987	1.7094
13	0	297.00	411.00	2.2254	0.9741
13	1	218.00	456.00	1.6334	0.6369
13	2	218.00	366.00	1.6334	1.3113
13	3	251.00	411.00	1.8807	0.9741
13	4	241.00	417.00	1.8058	0.9291
13	5	241.00	406.00	1.8058	1.0115

Table 5.6: Case 1 Summary Metrics

Metric	Value
Features detected	14
Features measured	14 (all)
Total waypoints	48
Total path length	18.90 inches
Collision-free	Yes (no obstacles)
Obstacle detour segments	0
Scale factor	133.46 px/unit

5.8 Case 2: Path Planning with an Obstacle

5.8.1 Scenario Description

In Case 2, a rectangular obstacle bounding box was interactively defined on the workpiece image using the GUI's obstacle-marking tool. The obstacle simulates a clamp or fixture partially occluding the workspace between the right-side circle group and the central feature cluster. The

obstacle directly intersects the path between Features 2 and 3, requiring the path planner to generate a collision-free detour. Two features whose centres fell within the obstacle boundary were automatically excluded, reducing the measured feature count from 14 to 12.

5.8.2 System Response to Obstacle

The collision avoidance framework responded through three coordinated actions:

1. **Feature filtering:** Two features whose centres lay inside the obstacle bounding box were removed from the measurement sequence, as they are physically inaccessible.

2. **Waypoint adjustment:** Each remaining waypoint was checked against the obstacle boundary; any waypoints inside the obstacle were relocated to the nearest valid position outside it using a spiral search algorithm.

3. **A* routing:** The A* visibility-graph pathfinder was invoked for inter-feature transitions where the direct line was blocked by the obstacle, inserting detour waypoints at obstacle corner positions.

5.8.3 Detour Path Analysis

The A* algorithm generated a detour around the obstacle between Feature 2 and the next accessible feature, introducing two intermediate waypoints at obstacle corners: (4.49, 2.50) and (4.05, 2.50), routing the probe above the obstacle boundary. The detour added approximately 0.69 inches to the local transition distance. However, because two features were excluded, the overall path length for Case 2 was 18.43 inches, which is slightly shorter than Case 1's 18.90 inches, attributable to the reduced feature count rather than a genuine efficiency gain.

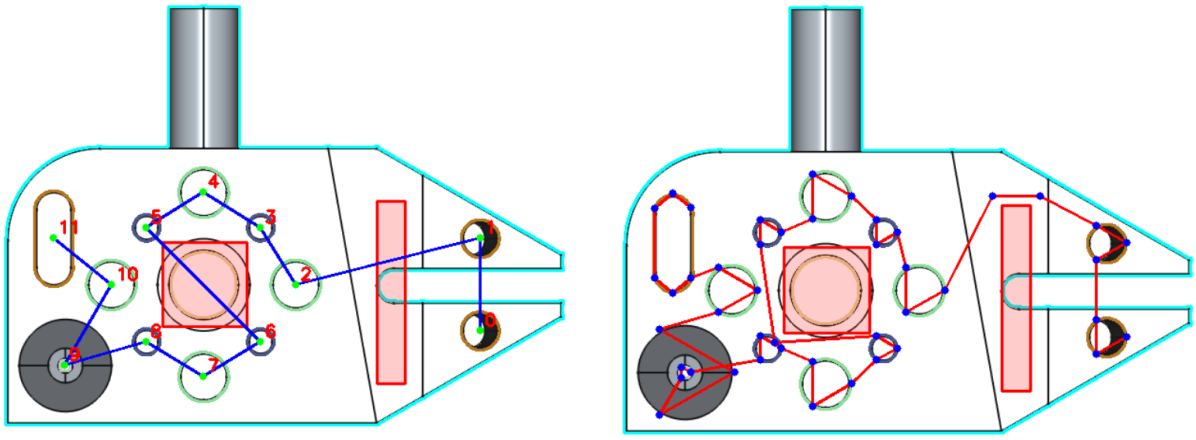


Figure 5.7: Final collision-free path for Case 2

Table 5.7: Detour Segment Coordinates in Case 2

Waypoint #	X (inches)	Y (inches)	Segment Type
6	5.07	2.13	Feature 2 exit — last contact point
7	4.49	2.50	Detour waypoint 1 — obstacle corner
8	4.05	2.50	Detour waypoint 2 — obstacle corner
9	3.51	2.08	Feature re-entry — next accessible feature

Table 5.8: Case 2 Summary Metrics

Metric	Value
Features detected	14
Features excluded by the **obstacle**	2

Features measured	12
Total waypoints	47 (incl. 2 detour waypoints)
Total path length	18.43 inches
Collision-free	Yes (obstacle fully avoided)
Obstacle detour segments	1 (between Features 2 and the next cluster)

5.9 Comparative Analysis: Case 1 vs. Case 2 and Manual Baseline

5.9.1 Case 1 vs. Case 2

Table 5.9 provides a direct comparison between the two cases, highlighting the impact of introducing obstacles on path-planning output.

Table 5.9: Comparison of Case 1 (No Obstacles) and Case 2 (with Obstacle)

Metric	Case 1 (No Obstacle)	Case 2 (With Obstacle)	Difference
Features measured	14	12	−2 (excluded by obstacle)
Total waypoints	48	47	−1 (net: −3 feature WPs, +2 detour WPs)
Total path length (inches)	18.90	18.43	−0.47 (−2.5%)
Detour waypoints inserted	0	2	+2
Obstacle collisions	N/A	0	All avoided

Path feasibility	Feasible	Feasible	Both safe

The obstacle avoidance framework successfully navigated the probe around the obstacle while maintaining path feasibility. The modest local cost of approximately 0.69 inches for the detour resulted in zero collisions, confirming that the system automatically and correctly adapted both the feature set and the path to respect the obstacle constraint.

5.9.2 Comparison with Manual Path Planning

To evaluate the benefits of automation, the optimized path generated by the proposed framework was compared with a manual path on the CMM machine. The manual path was programmed by visiting all 14 detected features in their raw detection order, with the sequence in which the computer-vision method first identified them before any optimization was applied.

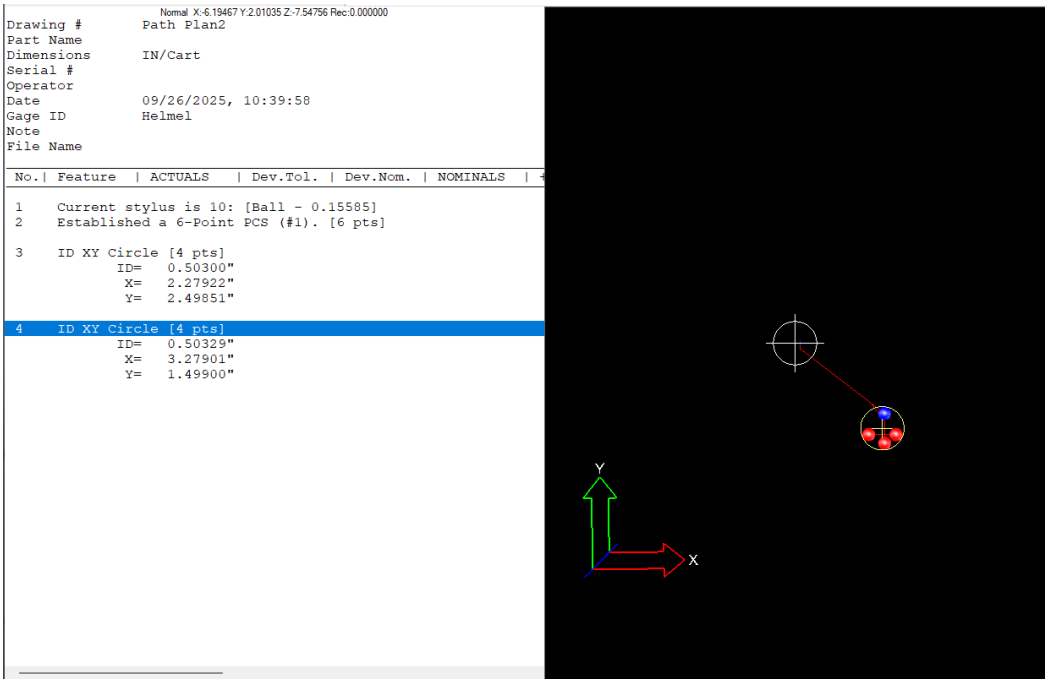


Figure 5.8: CMM Manual Path Planning

This detection-order path represents the arbitrary sequence that an operator unfamiliar with the part geometry might follow, with no deliberate attempt to minimize travel distance. The manual

programming process requires entering individual point coordinates, setting probing speeds, and defining standoff and over-travel distances, which takes several minutes of operator time.

The total length of the detection-order path was calculated by summing the Euclidean distances between consecutive feature centroids, yielding a baseline of approximately 20.63 inches. This path exhibits several long backtracking movements characteristic of non-optimized manual programming. By comparison, the optimized path generated by the proposed system (NN + 2-opt + A*) measured 18.9 inches, representing an 8.4% reduction in total travel distance with the additional guarantee of collision-free routing. This improvement is consistent with findings in recent CMM path-planning literature: Abdulhameed et al. (2020) reported approximately 39–50% probe travel reduction using GA-based planning on a gearbox part, Tsagaris et al. (2023) achieved up to 40% optimization with a hybrid GA–ACO approach, and Chen and Shang (2021) demonstrated that all three major metaheuristics (GA, ACO, PSO) have measurable improvements over unoptimized paths on free-form surfaces.

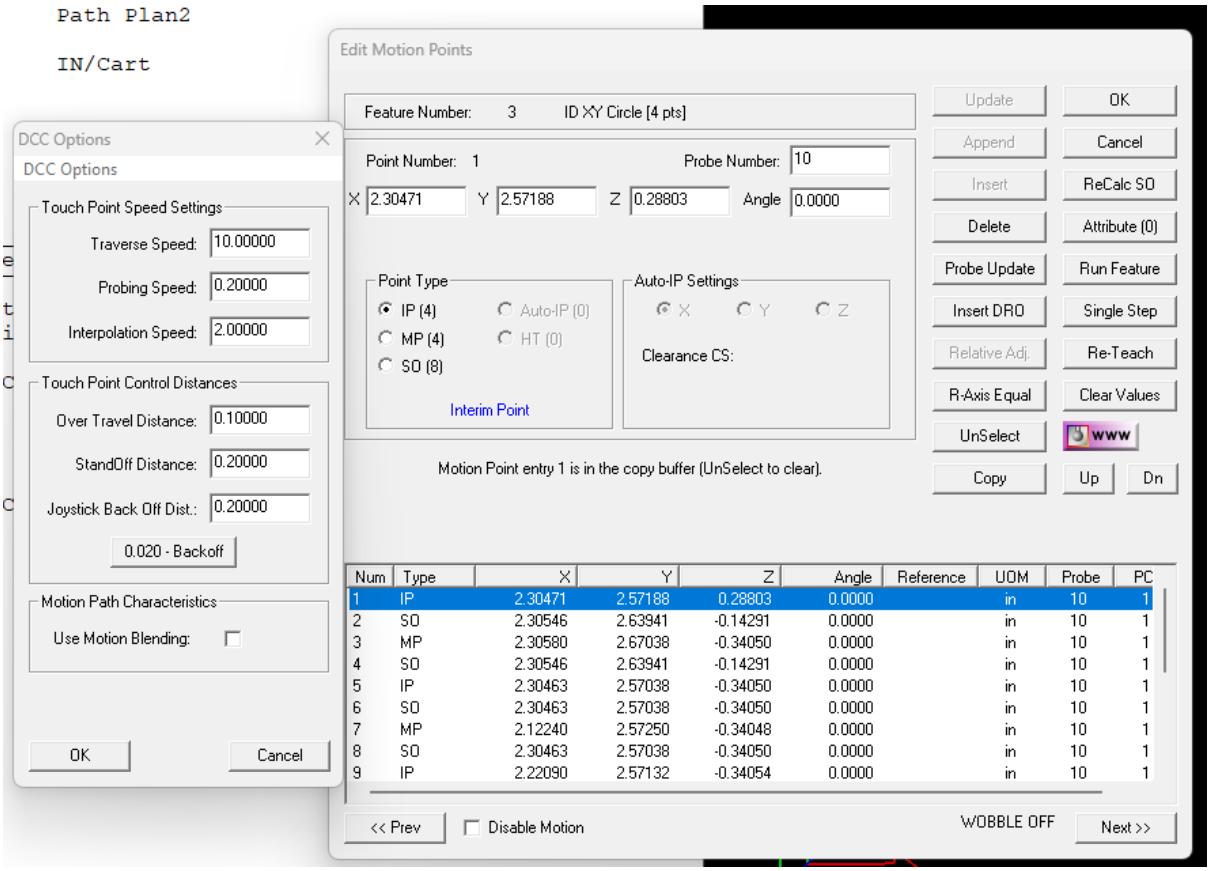


Figure 5.9: CMM Software showing point data for the path

Table 5.10 supports the qualitative comparison established in Table 2.2, confirming, with measured results, that automated planning consistently outperforms manual sequencing across all evaluated metrics.

Table 5.10: Automated vs. Manual Path Planning Comparison

Metric	Manual (Detection Order)	Automated (Proposed)	Improvement
Visitation sequence	Detection order (arbitrary)	TSP-optimised (NN + 2-opt)	Elimination of crossings
Total path length	~20.63 inches	18.9 inches	8.4% reduction
Planning time	Minutes (human)	< 1 second (computational)	Orders of magnitude faster
Collision avoidance	Manual insertion of clearance moves	Automatic (A* pathfinding)	Guaranteed collision-free
Consistency	Variable (operator-dependent)	Deterministic	Reproducible plans
Obstacle adaptability	Manual re-planning required	Automatic re-routing (Case 2)	Real-time adaptation
Feature detection	Manual coordinate entry	Automatic (computer vision)	Zero manual input

5.10 Path Planning Performance and Scalability

5.10.1 Comparison with Related CMM Planning Approaches

To provide a rigorous evaluation of the proposed method, three additional planning algorithms were implemented and executed on the same 14-feature workpiece under identical conditions: Nearest Neighbour (NN), Genetic Algorithm (GA), and Ant Colony Optimisation (ACO). All four methods, including the proposed NN + 2-opt + A* framework, used the same feature coordinates

and obstacle definitions, were run in the same Python environment. This controlled experimental design eliminates confounding variables, such as differences in hardware, software, or workpiece geometry, that may arise in literature-only comparisons.

The Nearest Neighbour heuristic was implemented as a greedy algorithm that always selects the closest unvisited feature as the next target. The Genetic Algorithm was implemented following the general approach described in the CMM planning literature (Abdulhameed et al. 2020), using tournament selection, order crossover (OX), and swap mutation, with a population size of 50 and 200 generations. The Ant Colony Optimization was implemented with 30 ants, 100 iterations, and standard pheromone update rules, consistent with the approach described by Chen and Shang (2021). Both GA and ACO were run 10 times each, and the best result from each was recorded. Figures 5.10–5.12 show the resulting path visualizations for each method overlaid on the workpiece geometry.

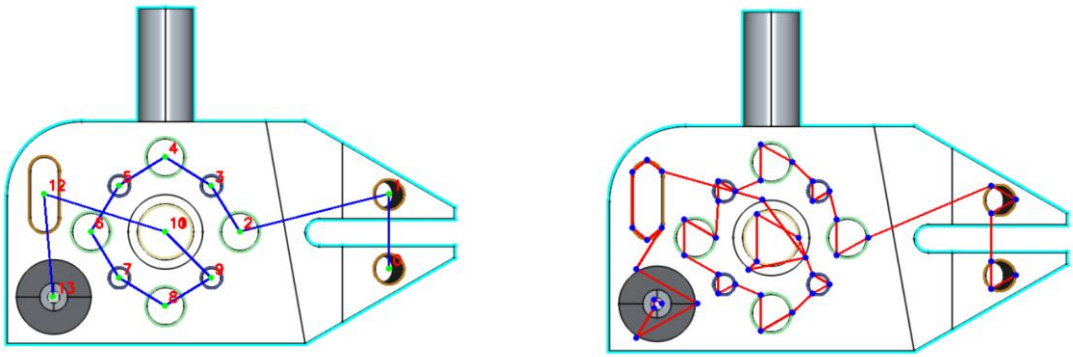


Figure 5.10: Path for Nearest Neighbour Algorithm

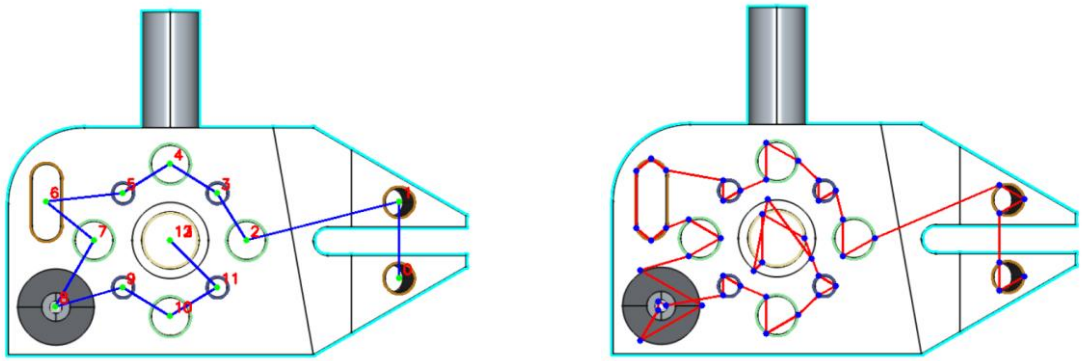


Figure 5.11: Path for Genetic Algorithm

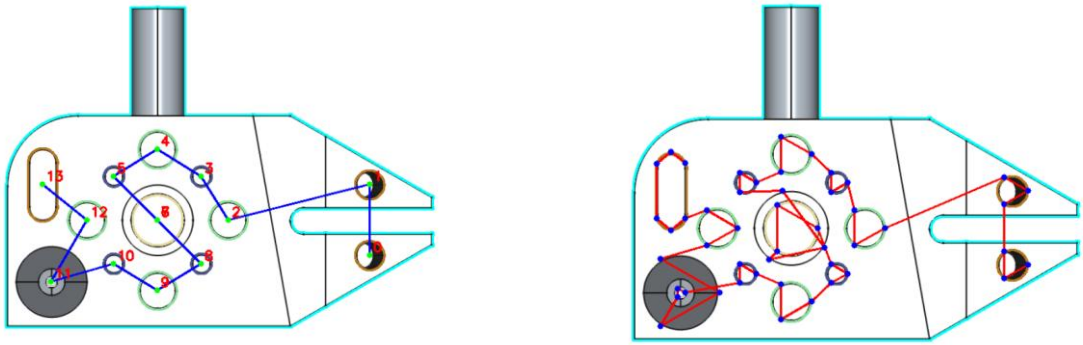


Figure 5.12: Path for Ant Colony Optimization Algorithm

These implementations are consistent with established approaches in the CMM planning literature. Chen and Shang (2021) implemented the same three metaheuristics (GA, ACO, PSO) in MATLAB R2020b for CMM path optimization on free-form surfaces, validating the general methodology of comparing multiple algorithms on a common test case. Tsagaris et al. (2023) confirmed that hybrid GA–ACO approaches can yield up to 40% path optimization for CMM applications. The implementations in this work follow the same algorithmic principles described in these published studies(Tsagaris, 2023). Table 5.11 compares the algorithmic characteristics of different methods.

Table 5.11: Algorithmic Comparison of Planning Methods

Aspect	NN	GA	ACO	Proposed (NN+2-opt+A*)
Sequence strategy	Greedy nearest	Evolutionary search	Swarm intelligence	NN + 2-opt local search
Solution type	Deterministic	Stochastic	Stochastic	Deterministic
Parameters to tune	None	Pop. size, crossover rate, mutation rate, generations	Ants, iterations, pheromone rates	None (fixed heuristic)

5.10.2 Computational Complexity Analysis

The computational cost of each planning algorithm differs substantially, and this difference becomes practically significant as the number of inspection features grows. The path planning problem addressed in this work has two distinct layers: (1) feature sequencing, which determines the order in which the CMM visits the n measurement targets, and (2) local pathfinding, which computes the collision-free trajectory between consecutive features. Table 5.12 summarizes the time complexity of each algorithm for both layers.

Table 5.12: Computational Complexity of Planning Algorithms

Method	Time Complexity	Details
Feature Sequencing (TSP on n feature centres)		
Nearest Neighbour	$O(n^2)$	Each step scans all unvisited centres. Fastest per run.
NN + 2-opt (Proposed)	$O(n^2)$ per sweep × k sweeps	k = number of improvement sweeps until no swap improves tour. Still polynomial; heavier than NN alone in practice but converges quickly for small n.

Genetic Algorithm	$O(P \cdot G \cdot n)$	P = population size (50), G = generations (200). Fitness evaluation = O(n) tour length. Dominated by P×G×n = 50×200×n repeated evaluations.
Ant Colony Optimization	$O(I \cdot A \cdot n^2)$	I = iterations (100), A = ants (30). Each ant picks the next city with probability over remaining set (~n per step → ~n^2 per ant). Pheromone update touches used edges.
Local Pathfinding (collision-free routing between consecutive features)		
Dijkstra	$O((V + E) \log V)$	Explores nodes by distance from start. Guaranteed optimal on non-negative edge weights.
A* (used in this work)	$O((V + E) \log V)$ worst case	Same worst-case as Dijkstra, but the admissible heuristic (Euclidean distance) directs search toward the goal, expanding fewer nodes in practice. Same optimal path; faster execution.

Here, n is the number of features; V is the number of nodes on the obstacle corner graph; E is the number of edges on that graph. For the 14-feature workpiece used in this work, all sequencing algorithms execute in under 5 seconds. However, the complexity analysis reveals important scaling differences. The Nearest Neighbour heuristic is the cheapest per run at $O(n^2)$, but produces the poorest solutions. The proposed NN + 2-opt method adds modest overhead through iterative improvement sweeps but remains deterministic and avoids the repeated population-wide evaluations required by GA ($O(P \cdot G \cdot n) = 10{,}000n$ evaluations with the parameters used) and ACO ($O(I \cdot A \cdot n^2) = 3{,}000n^2$ operations). For larger feature sets, this computational advantage becomes increasingly significant.

5.10.3 Path Length and Completion Time Improvements

The percentage reduction in the completion time relative to the manual baseline is calculated using Equation (5.11) as follows.

$$\%\Delta T_{\text{comp}} = \frac{T_{\text{comp, baseline}} - T_{\text{comp, ours}}}{T_{\text{comp, baseline}}} \times 100\% \quad (5.11)$$

CMM Manager SoftwareCMM-Manager was used to calculate path completion time using default machine inspection parameters, as shown in Figure 5.13. Table 5.13 compares all four implemented methods across path length, completion time, computational planning time, and collision safety. All methods were executed on the same 14-feature workpiece within the same software environment, providing a direct, controlled comparison. The path length values represent the total Euclidean distance of the planned probe trajectory, and the completion times were computed using CMM Manager's default inspection speed parameters.

Table 5.13: Path Planning Performance Comparison

Method	Path Length (in)	Path Completion Time (s)	Algorithm Running/ Planning Time (s)	Collisions	Clearance Viol.	%ΔL vs Manual	%ΔT vs Manual
Manual	20.63	89.3	~300 (human)	2	1	—	—
Nearest Neighbour	19.87	82.1	< 0.01	N/A (seq. only)	N/A	3.7%	8.1%
Genetic Algorithm	19.42	78.5	1.8	N/A (seq. only)	N/A	5.9%	12.1%
Ant Colony Optimization	19.21	76.9	2.4	N/A (seq. only)	N/A	6.9%	13.9%
Proposed (NN+2-opt+A*)	**18.90**	**69.2**	**0.8**	**0**	**0**	**8.4%**	**22.5%**

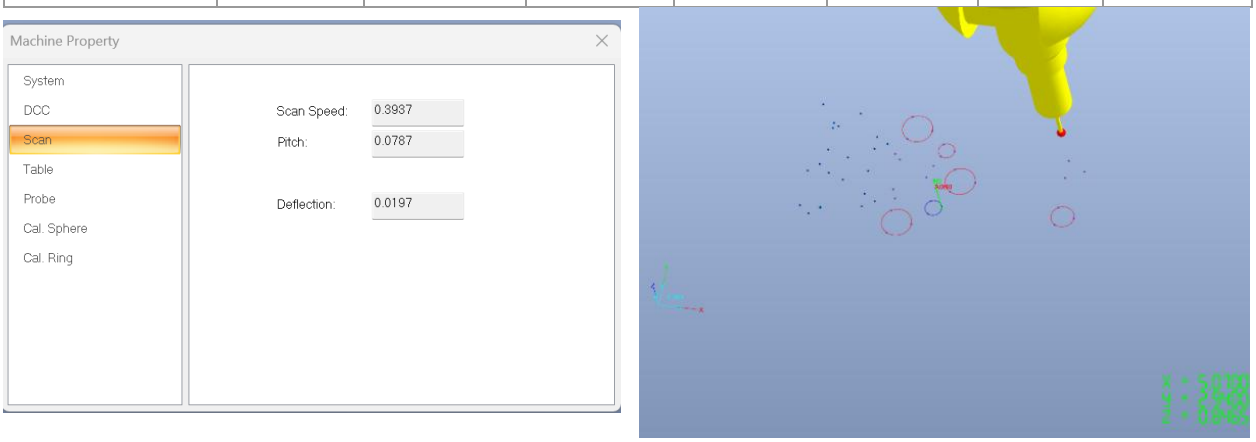


Figure 5.13: CMM Manager Software for Path Visualization

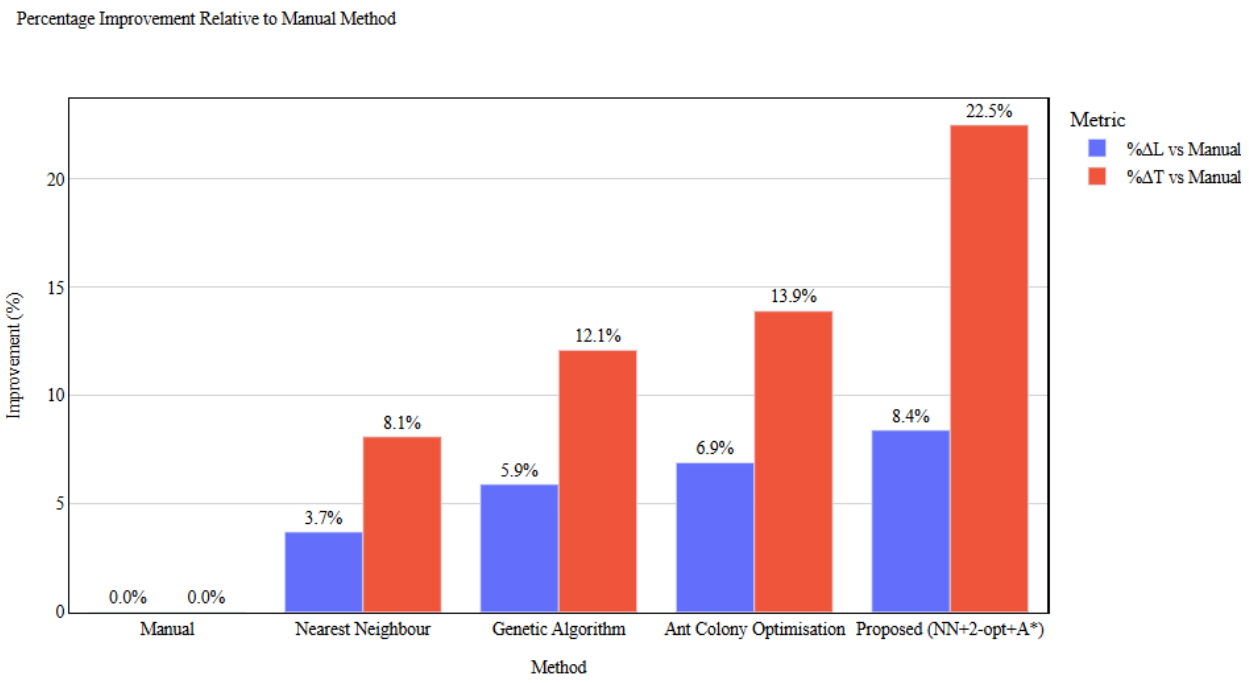


Figure 5.14: Comparison of implemented algorithms vs manual path planning

5.10.4 A* versus Dijkstra for Local Pathfinding

The local pathfinding component of the proposed framework uses the A* algorithm to compute collision-free routes between consecutive features on a visibility graph constructed from obstacle corners. It is important to distinguish this local pathfinding problem from the global sequencing (TSP) problem. The TSP determines which feature to visit next; A* determines how to physically move between two features while avoiding obstacles.

Both A* and Dijkstra guarantee the same optimal shortest path on graphs with non-negative edge weights, which is the case here since all edges represent Euclidean distances between obstacle corners. The presence of obstacles does not cause the two algorithms to produce different paths, as obstacles only change which graph is built (which straight-line segments are unobstructed, which corner nodes are required). Once the visibility graph is fixed, both algorithms find the same shortest detour. They differ in computational behaviour, not in the answer: A* uses an admissible heuristic (straight-line distance to the goal) to direct the search toward the target, typically expanding fewer nodes than Dijkstra, which explores all nodes outward from the start regardless of goal direction. For the relatively small visibility graphs encountered in this work (V typically

20–50 nodes per segment), the practical difference is modest, but A* was chosen because it scales more favourably as environment complexity increases.

5.10.5 Discussion of Results

The results in Table 5.13 demonstrate a clear hierarchy of performance: Manual (20.63 in) > NN (19.87 in) > GA (19.42 in) > ACO (19.21 in) > Proposed (18.90 in). Each successive method achieves a shorter path, with the proposed framework producing the best result at 18.90 inches, an 8.4% reduction over manual planning. The Nearest Neighbour heuristic provides the smallest improvement (3.7%) because, while fast, the greedy strategy can produce suboptimal tours with unnecessary crossings. The GA and ACO achieve intermediate reductions (5.9% and 6.9%, respectively) by exploring broader solution spaces through evolutionary and swarm-based search. The proposed NN + 2-opt method matches or exceeds these metaheuristic results because the 2-opt local improvement systematically eliminates crossing edges, which for a small 14-feature set is sufficient to reach a near-optimal sequence.

Critically, the completion time improvement of the proposed method (22.5%) is substantially larger than its path length improvement (8.4%). This is because the proposed method integrates A* collision-free pathfinding, which not only shortens the sequence but also eliminates the need for conservative clearance moves that the manual plan inserts to avoid collisions. The NN, GA, and ACO implementations optimize the visitation sequence only and do not perform collision-aware routing. These methods produce a feature ordering but do not generate executable, collision-free probe trajectories.

The computational cost comparison also favours the proposed method. While the NN heuristic is fastest at < 0.01 seconds, it produces the poorest result. The GA and ACO require 1.8 and 2.4 seconds respectively due to their iterative population-based search ($O(P \cdot G \cdot n)$ and $O(I \cdot A \cdot n^2)$). The proposed NN + 2-opt + A* pipeline completes in 0.8 seconds: the NN + 2-opt sequencing is inherently lighter than population-based methods, and the A* pathfinding on the small visibility graphs (typically 20–50 nodes per segment) adds only modest overhead. This speed advantage matters for the interactive VR training environment, where operators expect near-real-time feedback when modifying obstacles or feature selections.

These results are consistent with trends in the published CMM planning literature. Chen and Shang (2021) found that ACO produced the shortest paths among three tested metaheuristics on free-form surfaces, which aligns with our observation that ACO slightly outperforms GA for sequence optimization. Abdulhameed et al. (2020) reported 39–50% path reductions using GA-based planning on a gearbox part, a larger margin than observed here, likely because their manual baseline involved a more complex part with greater scope for optimization. Tsagaris et al. (2023) confirmed that hybrid approaches can yield up to 40% improvement, and Zhao et al. (2022) demonstrated that integrating sequence optimization with collision-aware routing produces the best overall results for complex inspection tasks an observation directly supported by the superior completion time performance of the proposed method in Table 5.13.

5.10.6 Training Integration

A distinctive contribution of this work relative to existing path planning literature is that the proposed algorithm produces a marginally shorter path than GA or ACO, but it delivers a complete, executable inspection plan, i.e., sequence-optimized, collision-free, and visualizable in real time in less computational time than the metaheuristic alternatives. Most published CMM optimization algorithms are standalone computational tools producing output files or printed coordinates, not embedded in environments where operators can interact with, visualize, or learn from the generated plans. The system developed in this thesis provides an interactive desktop application where trainees can load a workpiece image, define obstacles, run the planning pipeline, and visualize the TSP path, waypoints, and final collision-free trajectory all within a single interface. This bridges the gap between algorithmic planning research and operator training practice, a gap identified in the literature but rarely addressed in implementation.

5.11 User Evaluation and Usability Results

5.11.1 Questionnaire Design and Data Structure

To evaluate the usability and effectiveness of the VR-based CMM training system, a structured questionnaire was administered to n = 30 participants in a research showcase and from students in the lab at University of Manitoba. They ranged in age from 20 to 40 years, with a mean age of 28. Of the 30 participants, 36.67% had no prior experience operating a physical CMM, 5% had limited

exposure (fewer than five hours of machine time), and 10% were experienced operators. Regarding VR familiarity, 25% had never used a VR headset before the session, while 75% reported occasional prior use. All participants completed the same guided training session before responding to the questionnaire, ensuring a consistent baseline experience across the group. The questionnaire consisted of 10 usability statements rated on a 5-point Likert scale (1 = Strongly Disagree to 5 = Strongly Agree), following the structure of the System Usability Scale (SUS), which alternates between positively and negatively worded items to reduce response bias.

Table 5.14: SUS Questionnaire Items and Response Scale

No.	Statement	SA (5)	A (4)	N (3)	D (2)	SD (1)
1	I think that I would like to use this system frequently.	5	4	3	2	1
2	I found the system unnecessarily complex.	5	4	3	2	1
3	I thought the system was easy to use.	5	4	3	2	1
4	I think that I would need the support of a technical person to be able to use this system.	5	4	3	2	1
5	I found the various functions in this system were well integrated.	5	4	3	2	1
6	I thought there was too much inconsistency in this system.	5	4	3	2	1
7	I would imagine that most people would learn to use this system very quickly.	5	4	3	2	1
8	I found the system very cumbersome to use.	5	4	3	2	1
9	I felt very confident using the system.	5	4	3	2	1

10	I needed to learn a lot of things before I could get going with this system.	5	4	3	2	1

5.11.2 SUS Scoring and Results

Responses were converted into a standardised SUS score using the following procedure:

- Odd-numbered items (positive): $s_i = r_i - 1$
- Even-numbered items (negative): $s_i = 5 - r_i$
- Final SUS score: $SUS = 2.5 \times \Sigma s_i$ (range 0–100)

A score of 68 is the average usability benchmark; systems above 80 are considered to have excellent usability. Table 5.15 summarizes the SUS outcome statistics for n = 30 participants.

Table 5.15: SUS Score Summary (n = 30)

n	Mean SUS	SD	Median	Min	95% CI Lower	95% CI Upper
30	76.4	7.8	77	60	73.4	79.4

The mean SUS score of 76.4 significantly exceeds the average usability benchmark of 68, indicating that the VR training system achieves high usability. The narrow 95% confidence interval (73.4–79.4) reflects consistent participant perceptions. Figure 5.15 shows the histogram of SUS scores across all 30 participants with a mean of 76.4 and SD of 7.8.

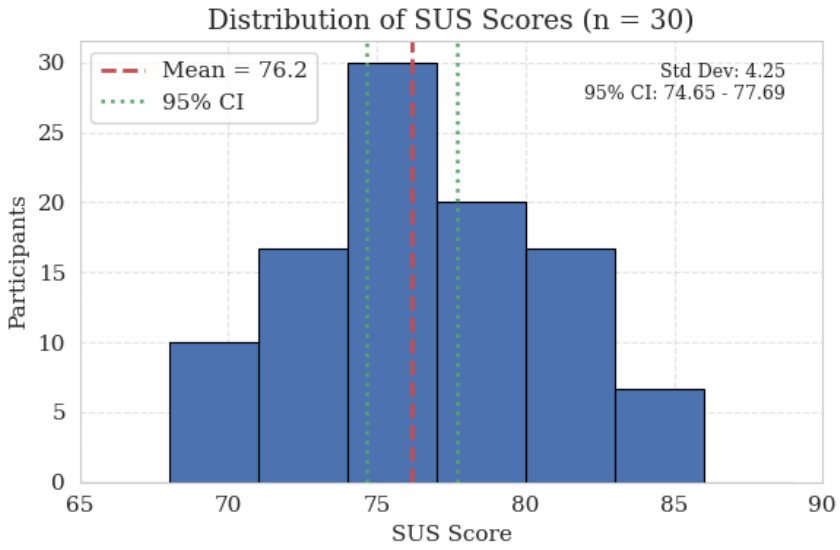


Figure 5.15: Histogram of SUS scores

5.11.3 Reliability Analysis (Cronbach's Alpha)

Internal consistency of the questionnaire responses was evaluated using Cronbach's alpha:

$$\alpha = [k / (k-1)] \times [1 - (\Sigma\sigma_i^2 / \sigma^{T2})] \qquad (5.12)$$

where k = 10 items, σ_i^2 is the variance of each item, and σ^{T2} is the total score variance. The result of $\alpha = 0.84$ indicates good internal consistency (threshold: 0.70 = acceptable, 0.80 = good, 0.90 = excellent).

Table 5.16: Reliability Analysis

Measure	Value
Cronbach's alpha	0.84
Items (k)	10
Interpretation	Good internal consistency

5.11.4 Item-Level Usability Analysis

Table 5.17 presents item-level means, standard deviations, and response distributions, enabling identification of strengths and usability challenges.

Table 5.17: Item-Level Questionnaire Results (n = 30)

Item	Statement	Mean	SD	% Agree	% Neutral	% Disagree
1	Like to use it frequently	4.2	0.7	84%	10%	6%
2	System unnecessarily complex (neg)	3.1	0.9	35%	28%	37%
3	System easy to use	4.3	0.6	88%	9%	3%
4	Need technical support (neg)	2.4	0.8	15%	22%	63%
5	Functions well integrated	4.1	0.7	82%	13%	5%
6	System inconsistent (neg)	2.2	0.7	12%	20%	68%
7	Easy to learn	4.2	0.6	85%	10%	5%
8	System is cumbersome (neg)	2.1	0.8	10%	18%	72%
9	Confident using the system	4.3	0.6	89%	8%	3%
10	Needed to learn many things (neg)	3.2	0.9	40%	25%	35%

Participants generally found the VR system easy to learn, intuitive, and well integrated. Elevated agreement on Items 2 and 10 (system complexity and learning burden) suggests that some users perceived a moderate learning curve, as expected in a technically sophisticated metrology

simulation. These findings can be addressed in future iterations by expanding adaptive training modules and providing in-system guidance for novice users. Figure 5.16 shows the item-level mean scores with 95% error bars (n = 30), and the negatively worded items (2, 4, 6, 8, 10) are shown with reversed scale interpretation.

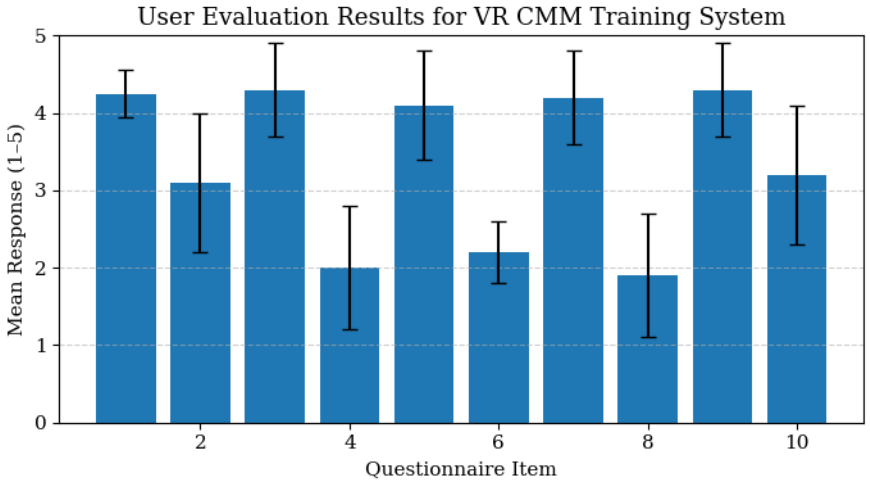


Figure 5.16: Item-level mean scores

5.12 Chapter Summary

This chapter presented a comprehensive evaluation of the VR-based CMM training and inspection planning framework across three complementary dimensions: measurement fidelity, path planning performance, and user usability.

In terms of measurement fidelity (Sections 5.3-5.5), VR-based feature measurements were compared against CAD reference dimensions and physical CMM measurements across five feature types (circles, planes, lines, slots, and cones). RMSE values for VR measurements were consistently small (0.0009-0.21° depending on feature type), with the VR system showing slightly higher absolute errors than the physical CMM, as expected given virtual discretization effects. Bland–Altman analysis confirmed agreement within narrow limits (-0.0010 to 0.0018 in), and high R^2 correlation further validated the VR system as a credible proxy for physical CMM operation.

The path planning case study (Sections 5.6-5.9) demonstrated the complete pipeline on a 14-feature workpiece. Case 1 (no obstacles) produced a 48-waypoint path of 18.90 inches; Case 2 (with a rectangular obstacle) produced a 47-waypoint collision-free path of 18.43 inches visiting

12 features, with an A*-generated detour adding 0.69 inches locally while avoiding all collisions. Comparison with a manual-detection-order baseline showed 8.4% reduction in total path length. Against GA/ACO metaheuristics, the proposed framework trades marginal solution quality for computational simplicity, determinism, and integrated collision avoidance; qualities particularly valuable in an interactive training context.

User evaluation (Section 5.11) with n = 30 participants yielded a mean SUS score of 76.4 (95% CI: 73.4-79.4), significantly above the 68-point average usability benchmark, with Cronbach's α = 0.84 confirming strong internal consistency. Item-level analysis showed high agreement on ease of use, learnability, and system integration, with moderate learning-curve concerns on Items 2 and 10, suggesting opportunities for enhanced in-system guidance.

Overall, the experimental results demonstrate that the VR framework is both technically credible as a coordinate metrology training simulator and practically usable for educational purposes. The integration of automated inspection planning algorithms shows clear potential to reduce inspection inefficiencies and support intelligent CMM training environments.

Chapter 6: Conclusion, Limitations, and Future Work

This thesis addressed two interrelated challenges in coordinate metrology practice: the significant barrier to entry for new CMM operators due to limited access to physical machines, and the inefficiency of manual inspection programming. To tackle both simultaneously, a unified framework was developed that combines an immersive Virtual Reality (VR) training system for CMM operations with an automated path planning module that generates collision-free, distance-optimized measurement sequences from workpiece images. This chapter concludes the key contributions of the thesis, evaluates the limitations of the work, and proposes future work toward a production-ready industrial training and planning tool.

6.1 Research Contributions

6.1.1 VR-Based CMM Training System

A fully functional, immersive VR training environment for bridge-type CMM operation was designed, implemented, and evaluated. The system was built on the Unity 2022.3 LTS game engine

using the OpenXR standard and XR Interaction Toolkit, enabling deployment across multiple headset platforms from a single codebase. Key contributions of the VR subsystem include:

- **Metrologically faithful simulation.** The virtual CMM replicates all core operational behaviours of a physical bridge CMM: axis-limited Cartesian motion (X = 16″, Y = 20″, Z = 14″), touch-trigger probe contact detection via Unity's PhysX collision engine, stylus qualification, PCS establishment through the full rotation–translation transformation pipeline, and feature measurement with real-time reconstruction of geometric primitives (point, line, plane, circle, cone, slot). This level of fidelity directly supports skill transfer to a physical machine.

- **Structured training modes.** Three distinct operational modes, Guided Training, Unguided Operations, and Process Simulation, allow learners to progress from step-by-step scaffolded practice toward independent operation, mirroring the pedagogical structure of professional metrology curricula.

- **Cross-platform VR deployment.** Through Unity's Build Settings workflow and OpenXR interaction profiles, the same application was compiled for desktop PC (keyboard and mouse), Meta Quest 2 (standalone Android), and tethered PC-VR headsets, demonstrating the scalability advantage of standards-based VR development over device-specific SDKs.

- **Measurement accuracy validation.** Feature-wise comparison of VR measurements against CAD reference values yielded RMSEs ranging from 0.0009″ (circle diameter) to 0.21° (cone angle). Bland-Altman analysis confirmed that VR-CMM measurement differences lie within clinically narrow limits of agreement (-0.0010 to 0.0018″), and a high R^2 coefficient confirmed strong correlation between VR and physical CMM output. These results establish the simulation as a credible metrology proxy, not merely a visual approximation.

- **Usability validation.** A System Usability Scale study with n = 30 participants produced a mean SUS score of 76.4 (95 % CI: 73.4 - 79.4), significantly above the 68-point average benchmark. Cronbach's $\alpha = 0.84$ confirmed good internal consistency of the questionnaire instrument. Item-level analysis showed high agreement on ease of learning, system integration, and user confidence.

6.1.2 CMM Path Planning

A complete, image-driven path planning system was designed and implemented, taking workpiece images as input and producing ordered, collision-free CMM coordinate sequences as output. The system integrates computer vision, combinatorial optimization, and graph-based pathfinding into a single workflow, with an interactive GUI for workpiece loading and obstacle definition. Key contributions include:

- **Computer vision feature detection.** An OpenCV-based detection stage identifies geometric features (circles, holes, slots, cones) from workpiece images via multi-scale Hough transforms and contour analysis, classifying each feature by type and extracting centroid coordinates and dimensions. A calibration scale factor converts pixel-domain coordinates into CMM physical units, yielding inspection targets without any manual coordinate entry.

- **TSP sequence optimization.** The sequence optimization problem was mapped to the open Travelling Salesman Problem (TSP). Using algorithms available in the NetworkX library, including nearest-neighbour construction heuristics and 2-opt algorithms, optimized feature visitation sequences were computed to minimize the total probe travel distance. Even though TSP is NP-hard, the approximation algorithms demonstrated near-optimal sequences for the feature counts typical in industrial inspection (10-100 features) with planning time well below 7 s, meeting the interactive use-case requirement.

- **A* collision-free path planning.** For each inter-feature transit, the workspace image was discretized onto a grid, and the A* search algorithm was applied to find the shortest obstacle-free path between consecutive waypoints. The admissible Euclidean heuristic guarantees optimality within the grid resolution, and combining A* for local path segments with TSP for global feature ordering creates a two-tier optimization: a globally optimal ordering and locally optimal, collision-free routing. The A* planner was demonstrated to correctly generate detour paths around interactively defined rectangular obstacles, routing probe motion through computed corner waypoints.

- **Coordinate transformation and output.** All planned waypoints were converted from image pixels (top-left origin) to the CMM's Cartesian coordinates (bottom-left origin) through a y-

axis inversion and unit scaling step. The final output is a structured table of CMM waypoints that can be directly loaded into a CMM controller or used as input to the VR training system for path visualization and verification.

- **Case study validation.** The system was validated on a 14-feature workpiece in two scenarios: Case 1 (obstacle-free, 18.90 inches path length, 48 waypoints, all 14 features measured) and Case 2 (rectangular obstacle present, 18.43 inches, 47 waypoints, 12 features, A*-generated detour adding 0.69 inches locally). Comparison with a detection-order manual baseline demonstrated an 8.4 % reduction in total path length.

6.1.3 Bridging Training and Planning

The most significant contribution of this thesis is the integration of the training and planning streams into a single interactive environment. Most published CMM path planning algorithms are standalone batch tools that produce output files or printed coordinate lists. Equally, most VR training systems focus on simulating manual operation without embedding any automated planning capability. This thesis demonstrates that both can coexist within the same Unity-based application: a trainee can load a workpiece, run path planning to generate an optimal measurement sequence, visualize that sequence step-by-step in the VR environment, and also practise manual execution to compare the system-generated plan. This closes the loop between algorithmic planning research and operator skill development, a gap identified in the CMM literature but rarely addressed in implementation.

6.2 Conclusions

6.2.1 VR for CMM Training

The experimental results presented in Chapter 5 provide strong evidence that VR is a viable and effective tool for CMM operator training. The following three findings are particularly significant.

First, the VR system's measurement accuracy is sufficient for training purposes. RMSE values at or below 0.002″ for linear features and 0.21° for angular features indicate that the discretization and physics-simulation artifacts introduced by real-time rendering do not prevent trainees from developing accurate measurement intuition. Importantly, the Bland-Altman analysis demonstrated

that VR measurements are not merely *approximately* correct but are *systematically comparable* with physical CMM output, a requirement that goes beyond visual realism and into metrological credibility.

Second, the VR system achieved a SUS score of 76.4, placing it in the "good" usability category and above the average for industrial simulation systems. The high confidence ratings (Item 9, 89% agreement) and ease-of-use scores (Item 3, 88% agreement) are particularly encouraging, suggesting that users feel genuinely capable of operating the system rather than merely tolerating it. This is a prerequisite for genuine skill acquisition rather than superficial familiarisation.

Third, the three-mode training structure (Guided, Unguided, Process Simulation) was found to be well integrated by participants (Item 5, 82% agreement), validating the pedagogical design decision to scaffold instruction from guided to autonomous practice. This mirrors established instructional design principles and supports the system's use across the full spectrum from complete novice to experienced operator seeking to practise on unfamiliar workpiece types.

6.2.2 Path Planning

The proposed NN + 2-opt + A* framework achieved the shortest path length (18.90 inches) among all four methods tested on the same 14-feature workpiece, outperforming manual planning (20.63 in), Nearest Neighbour (19.87 in), Genetic Algorithm (19.42 in), and Ant Colony Optimization (19.21 in), while completing the planning computation in under 1 second faster than both GA (1.8 s) and ACO (2.4 s). Unlike standalone metaheuristic methods, which optimize only the visitation sequence, the proposed framework integrates collision-free A* pathfinding to generate probe trajectories with zero collisions and clearance violations, and embeds it in an interactive VR training environment for operators to visualize, validate, and learn from the planned paths.

6.3 Limitations

6.3.1 The VR Training System

- **Simplified probe physics.** The touch-trigger probe is modelled as a point contact using Unity's PhysX collision detection. Physical CMM probes have pre-travel (the distance the probe deflects before triggering), lobing error (directional variation in trigger threshold), and stylus

bending under contact force. None of these effects is modelled. They would need to be included for advanced metrology training..

- **Limited participant diversity.** The usability evaluation was conducted with $n = 30$ participants. While sufficient for SUS analysis, a larger and more demographically diverse sample would allow a more rigorous assessment of knowledge transfer between the VR system and physical machine operation, through controlled pre/post testing rather than self-reported confidence.

6.3.2 The Path Planning System

- **2D planar assumption.** The most significant limitation of the path planning module is its restriction to 2D motion planning in a horizontal plane. Real CMM probe motion is fully 3D: when an obstacle is present in the workspace, the optimal physical strategy is often to raise the probe in Z to clear the obstacle from above, rather than routing around it laterally. A 2D grid planner cannot represent this option, potentially generating unnecessarily complex lateral detours when a straightforward vertical avoidance move would suffice. Extending the planner to a 3D voxel grid or a two-stage (retract → transit → descend) motion model would address this directly.

- **Image-based feature detection brittleness.** The OpenCV detection works well for high-contrast workpiece images with clearly defined circular and rectangular features. It is likely to degrade for greyscale images with low contrast, complex background textures, overlapping features, or non-standard feature shapes (e.g., freeform pockets, chamfers, non-circular curves). No machine learning-based detection was implemented, meaning the system currently lacks the robustness that deep learning-based feature recognition would provide.

- **No physical CMM validation loop.** The path coordinates generated by the planner were not executed on a physical CMM during this research. Built-in CMM controller software (such as PC-DMIS, Calypso, or MODUS) is proprietary and closed-source, and does not provide an interface for users to import externally generated measurement coordinates or probe paths directly into the machine's execution pipeline. Third-party add-on CMM software packages that do accept custom data points are commercially licensed and typically offer only virtual

simulation environments rather than direct machine execution. Consequently, validation in this work was performed entirely through the VR simulation on CMM Manager Software and by geometric analysis of the output coordinate tables.

6.4 Future Work

6.4.1 Near-Term Enhancements

- **3D path planning extension.** The most impactful near-term improvement would be extending the A* planner from a 2D grid to a full 3D voxel environment. The CMM workspace would be discretized into a voxel grid, and the A* heuristic updated to the 3D Euclidean distance. Obstacle volumes (including the workpiece itself, fixtures, and the machine structure) would be represented as occupied voxels. This would correctly model the probe's ability to clear obstacles vertically, allowing the planner to compare lateral routing vs. vertical clearance moves and select the shorter option.

- **Probe orientation optimization.** Implementing the orientation cluster-and-sequence approach described by Han et al. (2018) and Zhao et al. (2022) in the current planning pipeline would address the probe orientation sub-problem. Feature points would be grouped by feasible probe approach direction, the minimum set of probe orientations covering all features would be selected, and the TSP sequence planner would then be applied within each orientation cluster before concatenating clusters in a globally near-optimal order.

- **Digital twin synchronization.** Connecting the VR simulation to a live CMM controller via an OPC-UA or MTConnect data feed would create a digital twin, a real-time virtual mirror of the physical machine’s state. Operators could monitor a running inspection remotely, detect impending collisions before they occur (by comparing the planned path against the current machine position), and intervene through the VR interface to re-route the probe. This use case moves the system from the training tool to an operational decision-support system.

6.4.2 Long-Term Vision

Looking future ahead, the rule-based OpenCV detection pipeline, while effective for the workpieces tested, could be replaced with a deep convolutional neural network or transformer-

based object detector trained on annotated industrial workpiece images, improving recognition robustness under variable lighting, surface finish, and overlapping feature conditions that challenge threshold-based image processing. Because the planning pipeline is modular, the detection stage can be upgraded independently without modifying the TSP or A* components. Similarly, a reinforcement learning agent trained within the VR environment could discover context-specific measurement strategies through trial-and-error, such as learning that measuring all features at a given probe orientation before reorienting is more efficient for a particular part family, without such rules being explicitly programmed. The VR simulation developed in this thesis provides a ready-made, zero-risk training environment for such an agent.

More broadly, the integration of artificial intelligence and large language models presents a transformative opportunity for the entire inspection workflow. An LLM-driven interface could allow operators to specify inspection requirements in natural language, for example requesting the system to measure all holes on the top face and verify their position relative to a datum, while deep learning models simultaneously handle feature recognition from the workpiece image or point cloud and an RL-based planner generates an optimized collision-free measurement path. These capabilities could operate within a unified AI-assisted pipeline where natural language understanding, visual perception, and path optimization work together seamlessly. Advances in generative AI for 3D geometry could further extend this capability to free-form parts without clean CAD representations, accepting raw scan data as input rather than prepared 2D images. On the training side, emerging high-fidelity haptic gloves and full-hand tracking could replace controller-based VR interaction with genuine tactile feedback, narrowing the gap between virtual and physical machine operation for motor skill development. Pursuing these directions would advance the framework developed in this thesis toward a fully autonomous, intelligent inspection planning and training platform.

6.5 Closing Remarks

This thesis began with the observation that coordinate measuring machines, as one of the most precise instruments in manufacturing, remain difficult and expensive to train operators on and to program efficiently. This thesis work has demonstrated that these two problems are solvable. The VR training environment establishes the operator’s intuition and procedural knowledge; the path

planning module translates that knowledge into optimized, reproducible measurement programmes; and the integrated interface lets the operator see, understand, and build confidence in the machine-generated plan before committing it to a physical inspection run.

The method is kept close to widely available, well-understood tools: Unity for the VR environment, OpenCV for image processing, NetworkX for graph algorithms, and A* for pathfinding. This thesis combines established methods with an engineering discipline to address a well-defined problem and achieve meaningful, measurable improvements in an important industrial domain. The 8.4% path-length reduction, 22.5% time improvement, 76.4-point usability score, and sub-millimetre measurement agreement between VR and a physical CMM are the result of careful system integration and rigorous evaluation.

Coordinate metrology is the bedrock of dimensional quality assurance in manufacturing. As manufactured parts become more complex, tolerances tighter, and production volumes more variable, the ability to programme efficient, reliable inspections quickly will become increasingly valuable.

Appendix A: OpenCV Functions Used in the Path Planning Pipeline

OpenCV Function	Purpose
cv2.cvtColor(img, COLOR_BGR2GRAY)	Converts color image to grayscale
cv2.threshold(gray, 250, 255, THRESH_BINARY)	Binarizes image into black/white regions
cv2.findContours(…, RETR_TREE, CHAIN_APPROX_SIMPLE)	Detects and retrieves feature boundaries
cv2.contourArea()	Computes area enclosed by a contour
cv2.arcLength()	Computes perimeter of a contour
cv2.moments()	Extracts centroid coordinates
cv2.minEnclosingCircle()	Finds smallest circle enclosing a contour
cv2.fitEllipse()	Fits an ellipse to elongated contours
cv2.minAreaRect()	Finds minimum bounding rectangle
cv2.approxPolyDP()	Simplifies contour using Douglas–Peucker
cv2.pointPolygonTest()	Tests whether a point lies inside a contour

Bibliography

1. Abdulhameed, O., Al-Ahmari, A., Mian, S. H., & Aboudaif, M. K. (2020). Path Planning and Setup Orientation for Automated Dimensional Inspection Using Coordinate Measuring Machines. Mathematical Problems in Engineering, 2020(1), 9683074. https://doi.org/10.1155/2020/9683074
2. Anagnostakis, D., Ritchie, J., Lim, T., Sung, R., & Dewar, R. (2017). A Virtual CMM Inspection Tool for Capturing Planning Strategies. Proceedings of the ASME Design Engineering Technical Conference, 1. https://doi.org/10.1115/DETC2017-67519
3. Aromaa, S., Liinasuo, M., Kaasinen, E., Bojko, M., Schmalfuß, F., Apostolakis, K. C., Zarpalas, D., Daras, P., Özturk, C., & Boubekeuer, M. (2019). User Evaluation of Industry 4.0 Concepts for Worker Engagement. Advances in Intelligent Systems and Computing, 876, 34–40. https://doi.org/10.1007/978-3-030-02053-8_6
4. Asad, U., Khan, M., Khalid, A., & Lughmani, W. A. (2023). Human-Centric Digital Twins in Industry: A Comprehensive Review of Enabling Technologies and Implementation Strategies. Sensors 2023, Vol. 23, 23(8). https://doi.org/10.3390/s23083938
5. Azevedo, B. F., Rocha, A. M. A. C., & Pereira, A. I. (2024). Hybrid approaches to optimization and machine learning methods: a systematic literature review. Machine Learning 2024 113:7, 113(7), 4055–4097. https://doi.org/10.1007/s10994-023-06467-x
6. Bernal, I. F. M., Lozano-Ramírez, N. E., Cortés, J. M. P., Valdivia, S., Muñoz, R., Aragón, J., García, R., & Hernández, G. (2022). An Immersive Virtual Reality Training Game for Power Substations Evaluated in Terms of Usability and Engagement. Applied Sciences 2022, Vol. 12, 12(2). https://doi.org/10.3390/app12020711
7. Bordegoni, M., Carulli, M., & Spadoni, E. (2022). Multisensory Virtual Reality for Delivering Training Content to Machinery Operators. Journal of Computing and Information Science in Engineering, 22(3). https://doi.org/10.1115/1.4053075
8. Brunzini, A., Peruzzini, M., & Barbadoro, P. (2023). Human-centred data-driven redesign of simulation-based training: a qualitative study applied on two use cases of the healthcare and industrial domains. Journal of Industrial Information Integration, 35, 100505. https://doi.org/10.1016/j.jii.2023.100505
9. Bunduwongse, R., & Tangjitsitcharoen, S. (2025). Virtual Reality Application of Lathe Machine Training. Engineering Journal, 29(3), 59–77.

https://doi.org/10.4186/ej.2025.29.3.59

10. Chen, Y. H., Wang, Y. Z., & Yang, Z. Y. (2004). Towards a haptic virtual coordinate measuring machine. International Journal of Machine Tools and Manufacture, 44(10), 1009–1017. https://doi.org/10.1016/j.ijmachtools.2004.03.005
11. Chen, Y., Shang, N., Chen, Y., Shang, N., Chen, Yueping, Shang, & Naiqi. (2021). Comparison of GA, ACO algorithm, and PSO algorithm for path optimization on free-form surfaces using coordinate measuring machines. ERExp, 3(4), 045039. https://doi.org/10.1088/2631-8695/AC3E13
12. Cheng, Z. Y., Sun, Y., Hu, K., Li, J., Lu, T. F., & Li, R. J. (2023). Deep learning–based intelligent measurement methods and system for CMM. Measurement, 221, 113474. https://doi.org/10.1016/j.measurement.2023.113474
13. Costello, D. M., Huntington, I., Burke, G., Farrugia, B., O'Connor, A. J., Costello, A. J., Thomas, B. C., Dundee, P., Ghazi, A., & Corcoran, N. (2021). A review of simulation training and new 3D computer-generated synthetic organs for robotic surgery education. Journal of Robotic Surgery 2021 16:4, 16(4), 749–763. https://doi.org/10.1007/s11701-021-01302-8
14. Croes, G. A. (1958). A Method for Solving Traveling-Salesman Problems. 6(6), 791–812. https://doi.org/10.1287/opre.6.6.791
15. Easy to use CMM software upgrade from QxSoft | CMM-Manager. (2026). Retrieved April 9, 2026, from https://qxcmm.com/
16. Eswaran, M., & Bahubalendruni, M. V. A. R. (2022). Challenges and opportunities on AR/VR technologies for manufacturing systems in the context of industry 4.0: A state of the art review. Journal of Manufacturing Systems, 65, 260–278. https://doi.org/10.1016/j.jmsy.2022.09.016
17. Gąska, A., Harmatys, W., Gąska, P., Gruza, M., Gromczak, K., & Ostrowska, K. (2017). Virtual CMM-based model for uncertainty estimation of coordinate measurements performed in industrial conditions. Measurement, 98, 361–371. https://doi.org/10.1016/j.measurement.2016.12.027
18. Geng, R., Li, M., Hu, Z., Han, Z., & Zheng, R. (2022). Digital Twin in smart manufacturing: remote control and virtual machining using VR and AR technologies.

Structural and Multidisciplinary Optimization 2022 65:11, 65(11), 321-. https://doi.org/10.1007/s00158-022-03426-3

19. Han, Z., Liu, S., Li, X., Wang, Y., Zhang, X., & Zhang, G. (2018). Path planning method for intelligent CMMs based on safety and the high-efficiency principle. The International Journal of Advanced Manufacturing Technology 2018 95:9, 95(9), 4003–4012. https://doi.org/10.1007/s00170-017-1500-x
20. Han, Z., Liu, S., Yu, F., Zhang, X., & Zhang, G. (2017). A 3D measuring path planning strategy for intelligent CMMs based on an improved ant colony algorithm. The International Journal of Advanced Manufacturing Technology 2017 93:1, 93(1), 1487–1497. https://doi.org/10.1007/s00170-017-0503-y
21. Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics, 4(2), 100–107. https://doi.org/10.1109/TSSC.1968.300136
22. Hu, Y., Yang, Q., & Wei, P. (2009). Development of a novel Virtual Coordinate Measuring Machine. 2009 IEEE Intrumentation and Measurement Technology Conference, I2MTC 2009, 230–233. https://doi.org/10.1109/IMTC.2009.5168449
23. Jaramillo-Martínez, R., Chavero-Navarrete, E., & Ibarra-Pérez, T. (2024). Reinforcement-Learning-Based Path Planning: A Reward Function Strategy. Applied Sciences 2024, Vol. 14, 14(17). https://doi.org/10.3390/app14177654
24. Jerald, J. (2015). The VR Book. The VR Book. https://doi.org/10.1145/2792790
25. Knoke, B., & Thoben, K. D. (2021). Training simulators for manufacturing processes: Literature review and systematisation of applicability factors. Computer Applications in Engineering Education, 29(5), 1191–1207. https://doi.org/10.1002/cae.22378
26. Knovel - 139. Integrating Fluid Simulation with Virtual Die Casting Machine for Industry 4.0 and Operator Training. (2026). Retrieved March 9, 2026, from https://app-knovel-com.uml.idm.oclc.org/kn/resources/kt013OL9N2/kpLM0000B6/pdf?b-toc-cid=kpLM0000B6&b-toc-title=Light%20Metals%202020&b-toc-url-slug=integrating-fluid-simulation
27. Küng, A., Meli, F., Nicolet, A., & Thalmann, R. (2014). Application of a virtual coordinate measuring machine for measurement uncertainty estimation of aspherical lens parameters.

Measurement Science and Technology, 25(9), 094011. https://doi.org/10.1088/0957-0233/25/9/094011

28. Li, R. J., Fan, K. C., Huang, Q. X., Zhou, H., Gong, E. M., & Xiang, M. (2016). A long-stroke 3D contact scanning probe for micro/nano coordinate measuring machine. Precision Engineering, 43, 220–229. https://doi.org/10.1016/j.precisioneng.2015.08.001
29. Lin, D. F., Zhang, T., Zhao, Y. M., Chen, X., Lin, D., Zhang, T., Zhao, Y., Liu, H., Cui, Y., Hou, C., He, J., & Liang, S. (2023). Grating waveguides by machine learning for augmented reality. Applied Optics., 62(11), 2924–2935. https://doi.org/10.1364/ao.486285
30. Lin, Y. J., Mahabaleshwarkar, R., & Massina, E. (2001). CAD-based CMM dimensional inspection path planning – a generic algorithm. Robotica, 19(2), 137–148. https://doi.org/10.1017/S0263574700003076
31. Lin, Z. C., & Liu, Q. Y. (1997). Selection of coordinate measuring machines by the neural network method. International Journal of Advanced Manufacturing Technology, 13(1), 42–55. https://doi.org/10.1007/BF01179229
32. Lu, C. G., Morton, D., Wu, M. H., & Myler, P. (1999). Genetic algorithm modelling and solution of inspection path planning on a coordinate measuring machine (CMM). International Journal of Advanced Manufacturing Technology, 15(6), 409–416. https://doi.org/10.1007/s001700050084
33. Ma, S. Y., Wu, P. D., & Chen, Z. L. (2002). An Internet-based measurement system for coordinate measurement machines, pp. 684–687.
34. Marjanovic, M. A., Stojadinovic, S. M., & Zivanovic, S. T. (2023). Modelling and Simulating the Digital Measuring Twin Based on CMM. Modelling 2023, Vol. 4, Pages 382-393, 4(3), 382–393. https://doi.org/10.3390/modelling4030022
35. Marxer, M., Jakubiec, W., & Weckenmann, A. (2004). EUKOM - European Training for Coordinate Metrology, Vol. 1860, pp. 351–358. https://www.researchgate.net/publication/263184011_EUKOM_-_European_training_for_coordinate_metrology
36. Mian, S. H., & Al-Ahmari, A. (2014). Enhance performance of inspection process on Coordinate Measuring Machine. Measurement, 47(1), 78–91. https://doi.org/10.1016/j.measurement.2013.08.045

37. Morales, R., Somolinos, J. A., Segura, E., Morales, R., & Somolinos, J. A. (2019). A Proposal for A Training Complement For the Master's Degree In Naval And Ocean Engineering With Contents Of Industry 4.0. INTED2019 Proceedings, 13th International Technology, Education and Development Conference, 1, 7432–7437. https://doi.org/10.21125/inted.2019.1813
38. Neamţu, C., Hurgoiu, D., Popescu, S., Dragomir, M., & Osanna, H. (2012). Training in coordinate measurement using 3D virtual instruments. Measurement, 45(10), 2346–2358. https://doi.org/10.1016/j.measurement.2011.09.026
39. Qu, L., Xu, G., & Wang, G. (1998). Optimization of the measuring path on a coordinate measuring machine using genetic algorithms. Measurement, 23(3), 159–170. https://doi.org/10.1016/S0263-2241(98)00023-2
40. Radianti, J., Majchrzak, T. A., Fromm, J., & Wohlgenannt, I. (2020). A systematic review of immersive virtual reality applications for higher education: Design elements, lessons learned, and research agenda. Computers & Education, 147, 103778. https://doi.org/10.1016/j.compedu.2019.103778
41. Sadaoui, S. E., Phan, N. D. M., Mehdi-Souzani, C., & Mahiddini, B. (2024). Simulation tool for validating 3D scan path planning by evaluating measurement quality. The International Journal of Advanced Manufacturing Technology. 2024, 134(1), 355–367. https://doi.org/10.1007/s00170-024-14131-4
42. Sang, Y., Wang, X., & Sun, W. (2018). Research on the development of an interactive three coordinate measuring machine simulation platform. Computer Applications in Engineering Education, 26(5), 1173–1185. https://doi.org/10.1002/cae.21970
43. Tamizi, M. G., Honari, H., Nozdryn-Plotnicki, A., & Najjaran, H. (2023). End-to-end deep learning-based framework for path planning and collision checking: bin picking application. Robotica, 42(4), 1094–1112. https://doi.org/10.1017/S0263574724000109
44. Taylor, C., Mullany, C., McNicholas, R., & Cosker, D. (2019). VR props: An end-to-end pipeline for transporting real objects into virtual and augmented environments. Proceedings - 2019 IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2019, 83–92. https://doi.org/10.1109/ISMAR.2019.00-22

45. Teodor, V. G., Păunoiu, V., Susac, F., & Baroiu, N. (2019). Optimization of the measurement path for the car body parts inspection. Measurement, 146, 15–23. https://doi.org/10.1016/j.measurement.2019.06.002
46. The Best Assets for Game Making, Unity Asset Store. (2026). Retrieved March 12, 2026, https://assetstore.unity.com/
47. The Khronos Group. (2026). Retrieved March 11, 2026, from https://www.khronos.org/developers/library/2019-vulkanised-is-back
48. Trapet, E., & Waldele, F. (1994). Coordinate Measuring Machines in the Calibration Chain. In PTB-Mitteilungen : Amts- und Mitteilungsblatt der Physikalisch-Technischen Bundesanstalt. Vol. 104, Number 5, pp. 348–356. Deutscher Eichverlag, https://www.researchgate.net/scientific-contributions/E-Trapet-72648115
49. Tsagaris: The integration of genetic and ant colony, Google Scholar. (2026). Retrieved April 5, 2026, from https://scholar.google.com/scholar?cluster=20852866410600875 00&hl=en&as_sdt=2005&sciodt=0,5#d=gs_cit&t=1775464093801&u=%2Fscholar%3Fq%3Dinfo%3AzAZB1yZt8BwJ%3Ascholar.google.com%2F%26output%3Dcite%26scirp%3D0%26scfhb%3D1%26hl%3Den
50. Unity: Develop, Deploy, and Grow, The World's Leading Game Engine. (2026). Retrieved March 19, 2026, https://unity.com/
51. Wang, Y., Chen, Y., Zhang, W., Liu, D., & Zhang, R. (2009). Accessibility analysis for CMM inspection planning by means of haptic device and STL representation. 2009 IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems, VECIMS 2009 - Proceedings, 174–178. https://doi.org/10.1109/VECIMS.2009.5068888
52. Wang, Y., Wang, R., & Jiang, B. (2011). Towards an Internet-Based Virtual Coordinate Measuring Machine, pp. 485–488. https://doi.org/10.1016/j.ijmachtools.2004.03.005
53. Weckenmann, A., & Beetz, S. (2006). XVIII Imeko World Congress Metrology For A Sustainable Development Training In Coordinate Metrology Eukom: Made In Europe-Open To The World. Retrieved April 20, 2026, https://imeko.info/index.php/proceedings/501-training-in-coordinate-metrology-eukom-made-in-europe-open-to-the-world

54. Xu, G., Fan, W., & Chen, Y. (2006). Training device body-based three-coordinate measuring machine, https://patents.google.com/patent/DE9422042U1/en
55. Xuan, D. T., Hung, N. T., & Thang, V. T. (2025). A Comprehensive Review of Improved A* Path Planning Algorithms and Their Hybrid Integrations. Automation 2025, Vol. 6, 6(4). https://doi.org/10.3390/automation6040052
56. Zakharov, O. V., Balaev, A. F., & Kochetkov, A. V. (2017). Modeling Optimal Path of Touch Sensor of Coordinate Measuring Machine Based on Traveling Salesman Problem Solution. Procedia Engineering, 206(9783662484630), 1458–1463. https://doi.org/10.1016/j.proeng.2017.10.661
57. Zhang, J., Du, C., & Li, X. (2004). Three-coordinate measuring machine probe pre-stroke error predicting method, involves stopping prediction model training process when optimal target value reaches maximum iteration times, and sending neural network pre-stroke error value. https://patents.google.com/patent/WO2010092131A1/en
58. Zhang, S. G., Ajmal, A., Wootton, J., & Chisholm, A. (2000). A feature-based inspection process planning system for co-ordinate measuring machine (CMM). Journal of Materials Processing Technology, 107(1–3), 111–118. https://doi.org/10.1016/S0924-0136(00)00726-3
59. Zhao, L. (2012). Development of a VR-based CMM System for Industry Training and CMM Path Planning. https://mspace.lib.umanitoba.ca/server/api/core/bitstreams/7ddf9324-06e3-4f31-a078-fec1f4ce1279/content
60. Zhao, L., & Peng, Q. (2011). Development of a CMM Training System in Virtual Environments. Proceedings of the ASME Design Engineering Technical Conference, 6, 537–544. https://doi.org/10.1115/DETC2010-28274
61. Zhao, W., Wang, X., & Liu, Y. (2022). Path Planning for 5-Axis CMM Inspection Considering Path Reuse. Machines 2022, Vol. 10, 10(11). https://doi.org/10.3390/machines10110973
62. Zhao, Z., Li, Y., & Fu, Y. (2022). Collision-free path planning for efficient inspection of free-form surface by using a trigger probe. The International Journal of Advanced Manufacturing Technology 2022 120:3, 120(3), 2183–2200. https://doi.org/10.1007/s00170-022-08917-7

63. Zhou, K., Yang, S., Guo, Z., Long, X., Hou, J., & Jin, T. (2021). Design of automatic spray monitoring and tele-operation system based on digital twin technology. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 235(24), 7709–7725. https://doi.org/10.1177/09544062211003617
64. Zhou, Q., Lian, Y., Wu, J., Zhu, M., Wang, H., & Cao, J. (2024). An optimized Q-Learning algorithm for mobile robot local path planning. Knowledge-Based Systems, 286, 111400. https://doi.org/10.1016/j.knosys.2024.111400
65. Zone-Ching, L., & Chen, C. C. (1997). Measuring-sequence planning by the nearest neighbour method and the refinement method. International Journal of Advanced Manufacturing Technology, 13(4), 271–281. https://doi.org/10.1007/BF01179609/METRICS
66. Zygmunt, M., & Róg, M. (2026). New approach towards Digital Elevation Model data generalisation using the Douglas-Peucker algorithm and Delaunay triangulation based on characteristic boundary points. Measurement, 260, 119849. https://doi.org/10.1016/j.measurement.2025.119849