VIRTUAL AND PHYSICAL PROTOTYPING
2023, VOL. 18, NO. 1, e2276257 (27 pages)
https://doi.org/10.1080/17452759.2023.2276257

Taylor & Francis
Taylor & Francis Group

Online cooperative printing by mobile robots

Abdullah Alhijaily [a], Zekai Murat Kilic[a] and Paulo Bartolo[a,b]

[a]Department of Mechanical, Aerospace and Civil Engineering, The University of Manchester, Manchester, UK; [b]Singapore Centre for 3D Printing, School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore

ABSTRACT

Cooperative printing, where multiple printheads concurrently print a part, significantly improves printing speed. However, current literature only discussed offline path planning, in which the toolpaths are generated before the printing process starts. Offline path planning is unreliable and leads to collisions for systems with uncertainties such as mobile robots. In this paper, we developed several online path planning algorithms for cooperative printing by mobile robots that allow toolpath allocation in real time. Unlike offline path planning, it is not possible to replan the layer in case of collision in online systems. Thus, we developed a novel algorithm that guarantees collision avoidance in real time. The system was evaluated through both simulations and experiments. The mobile robots cooperatively printed several layers which showed that the system can significantly increase the speed of 3D printing. This work stands as the first in the literature that allows online path planning for cooperative printing.

ARTICLE HISTORY
Received 2 August 2023
Accepted 17 October 2023

KEYWORDS
3D printing; online path planning; mobile 3D printer; real-time path planning; cooperative printing

1. Introduction

The capabilities of printing systems can be significantly increased when printheads cooperate. In cooperative printing, multiple printheads work together concurrently to manufacture a single part. This concept leads to a substantial increase in production speed, enhancing overall throughput and efficiency [1]. Moreover, increasing the velocity of the printhead increases production speed and reduces printing time [2]. For any print velocity, n cooperative printheads can approximately finish the printing process at the same time when using $\frac{1}{n}$ the print velocity, which in turn reduces the required acceleration of the system allowing it to use less motor power. Furthermore, for each material there is a correlation between optimal printing velocities and final properties (e.g. the printing velocity determines the cooling process, which controls the crystallisation process and final mechanical properties) [3], increasing the print velocity to reduce the printing time is not always feasible.

In additive manufacturing (AM) gantry mechanical systems have dominated the market as they are easy to control and allow high accuracy and precision [4]. More recently, robotic arms have started experiencing a notable surge in adoption competing with gantry robots as they demonstrate enhanced versatility and allow multi-axis control [5,6]. An emerging mechanical system that is being considered for AM is the use of mobile robots.

Mobile robots are increasingly competing with fixed robotic systems by offering various improvements to AM [7]. Firstly, mobile 3D printers streamline portability and logistics. Fixed robots are designed to 3D print in their installed locations. If a new site is required, the machine must be disassembled, transported, and redeployed in the new location. Mobile robots, on the other hand, are portable, they can move to any location and start printing from there with minimal installation. Furthermore, mobile robots offer huge workspaces compared to fixed robots, which allows to print large-scale parts [8,9]. Moreover, mobile robots can reach places that are hard or impossible to reach by humans. For example, areas of disasters, underwater and in-space manufacturing [10]. Such places are difficult for humans to travel to and install a fixed machine to 3D print there.

Integrating mobile robots with cooperative printing synergistically enhances the capabilities of both mobile 3D printing and cooperative printing methodologies [1]. Utilising multiple mobile robots for large-scale

CONTACT Paulo Bartolo pbartolo@ntu.edu.sg Department of Mechanical, Aerospace and Civil Engineering, The University of Manchester, Manchester M13 9PL, UK; Singapore Centre for 3D Printing, School of Mechanical and Aerospace Engineering, Nanyang Technological University, 50 Nanyang Avenue, N3.1 B2C 03, Singapore 639798, Singapore

printing simplifies the complexity of tasks. Notably, travel motions, which are essentially preparatory moves for printing actions, can be reduced in both frequency and distance when printing responsibilities are shared among several robots. Large travel distances without active extrusion from the printhead can risk material solidification within the nozzle. Although mobile robots have huge workspaces, the entirety of the workspace cannot be fully utilised due to constraints like material supply, cable lengths, or communication range [1]. Therefore, deploying numerous cooperative robots streamlines design and lessens these limitations; a team of smaller robots offers greater flexibility than a singular large one [11]. Moreover, mobile robots are particularly well-suited for cooperative printing. Unlike multi-axis robotic arms, which often feature rigid links complicating collision detection and avoidance, mobile robots have enhanced movement capabilities in 2D – and in 3D for aerial variants – providing new collision avoidance strategies compared to their stationary counterparts.

The concept of cooperative printing improves the capabilities of AM but comes with its own challenges such as toolpath allocations, collision checking, and collision avoidance [1]. The main challenge for path planning of cooperative printing by mobile robots is that mobile robot's motion is noisy and exhibit uncertainties that may lead to the robots moving out of the planned toolpath. In this case, the path planner should consider the real-time motion of the mobile robot during the planning phase and adjust the toolpaths on the fly, instead of planning all the toolpaths beforehand. This online planning of cooperative systems was not explored before, and it is the focus of this work.

2. Related work

2.1. Cooperative printing systems

Various cooperative printing systems implementing material extrusion have been proposed [12–16]. Gantry systems were the first to be used as they allow various configurations for cooperative printing [17–21]. Zhang and Khoshnevis [22] developed a simulation that incorporates several algorithms to allow a 1.5 m width gantry to construct several structural objects. The success rate of finding a solution differs based on the algorithm used, but when a solution is found, the printing time reduction ranges from 18% to 49%. A physical cooperative gantry system was developed by Bui et al. [19], using two filament-based gantries to print a single layer plastic object. In this work, authors showed that their offline path planning for gantry systems outperforms other approaches used in commercial cooperative gantry systems such as Cronus [23] and Project Escher [24].

Contrary to gantry systems, robotic arms cooperative printing systems are easily expandable [17,25–27]. Shen et al. [28] used four robotic arms, with the capability of adding more, for plastic material extrusion and showed that the system is capable of reducing the printing time by up to 73%. Similarly, Bhat et al. [14] used three robotic arms for wire arc AM and developed an offline algorithm dedicated to finding the optimal placement of the three robotic arms that varies based on the geometry of the fabricated part.

Recently, mobile robots have been studied as an alternative to gantry and robotic arm systems for cooperative printing [1,7,12,15,16,29]. Sustarevas et al. [29] developed a mobile 3D printer designed for structural fabrication. In this system, an omnidirectional mobile robot carries a 5-axis robotic arm which includes a material extrusion system. However, the cooperation task between the mobile robots was carried out in simulation which showed that the system was able to construct large-scale objects. Similar virtual printing studies were performed by Zhang et al. [16] for cooperative aerial AM, mobile robots virtual printing by Poudel et al. [30], and cooperative printing for general robotic systems was simulated by Cai and Choi [17]. Nevertheless, several authors explored physical cooperative printing by mobile robots. For example, Xu et al. [15] used two non-omnidirectional mobile robots equipped with a filament-based material extrusion system to cooperative print a large-scale part. However, none of these systems implemented fully online cooperative path planning. An in-depth review of cooperative AM using teams of robots can be found in [1].

2.2. Cooperative printing path planning

Path planning for cooperative printing systems is implemented using three approaches (Figure 1): fully offline, semi-offline, and online. In all of the previously mentioned cooperative systems, path planning was either performed fully offline or semi-offline. In this context, 'online' (or real time) refers to path planning that occurs during the actual process, while 'offline' denotes path planning that is conducted prior to the process starts.

In fully offline path planning, toolpaths for all robots are predetermined prior to task execution, meaning that once these paths are set, they aren't updated

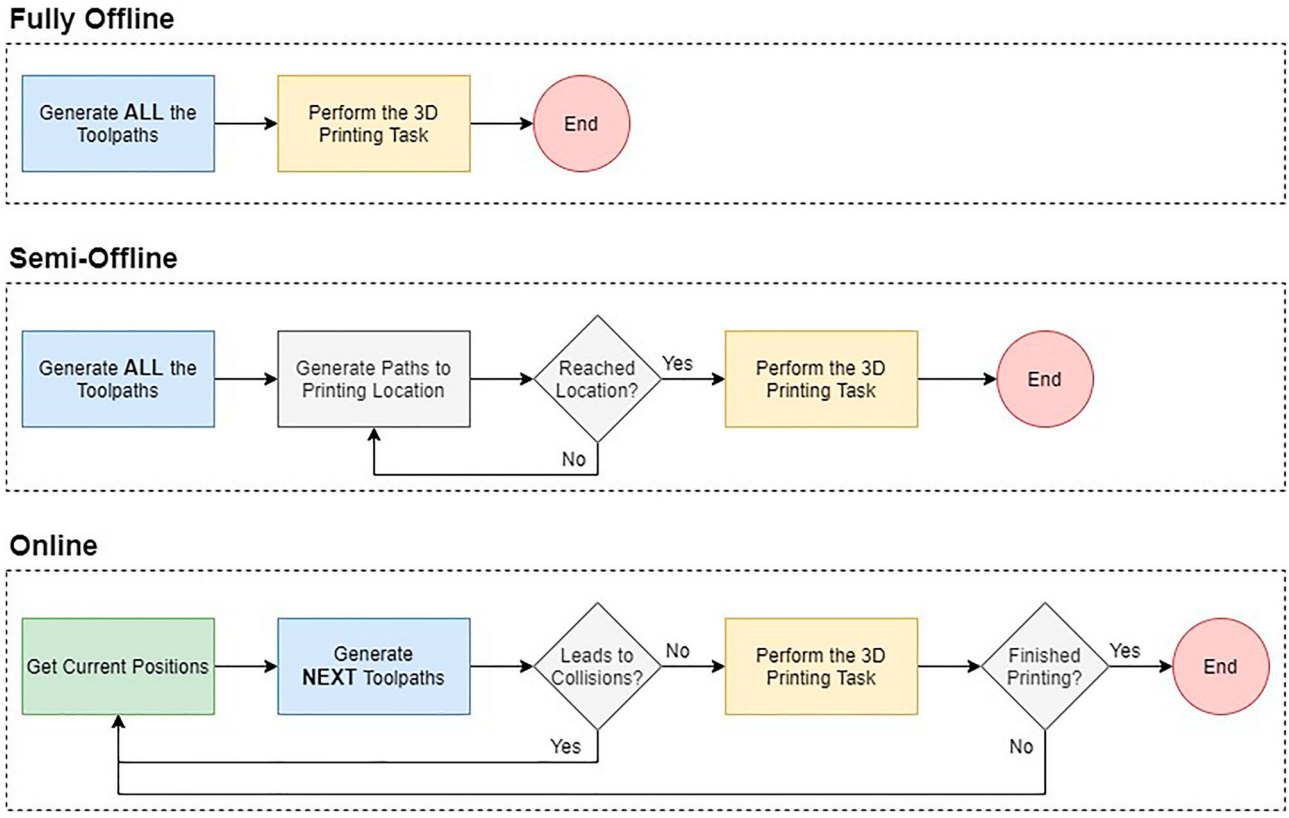

Figure 1. Generic flowchart of the three path planning approaches.

during the actual task [31]. The important feature of fully offline path planning is that performance is already known beforehand, allowing for multiple replanning sessions until the desired outcome is achieved. Predominantly employed for single-printhead systems, this path planning method is used in slicing software that generates a G-code file with all the instructions and toolpaths of the system [32,33]. However, fully offline path planning is also implemented in most cooperative printing systems. In all gantry and robotic arms cooperative printers presented in the literature, the implemented path planning was fully offline [14,18–20,22,25,28,34,35]. While inter-robot communication exists within offline planning, it's often limited to fundamental commands like initiating or terminating operations. Poudel et al. [12] showcased inter-robot communications in their team of cooperative mobile and robotic arms for swarm manufacturing. In this fully offline path planning system, basic predefined commands such as Go, Wait and Notify were allowed between the robots. However, these commands are already generated offline then fed to the robots within the generated G-code file. Fully offline path planning is better suited to systems with very high repeatability and accuracy such as gantry systems in which deviations rarely occur [1]. However, for dynamic systems in unpredictable environments, changing the predetermined toolpaths in real time is necessary, otherwise, collisions that were not accounted for in offline may occur. Another limitation of offline planning is its intolerance to unforeseen failures [11]. If a printhead malfunctions mid-process, the remaining printheads, bound by their predetermined toolpaths, cannot adapt to this unexpected complication, risking collisions with the faulty printhead. In this case, the solution is to halt the entire printing operation.

Semi-offline path planning implements both offline and online approaches. This approach is found exclusively in the literature of cooperative printing with mobile robots [9,30,36]. In this approach, the toolpath of each printhead is already predetermined in offline, however, the motion of the mobile robot from the docked location to the printing platform is performed online. Zhang et al. [9] in their cooperative printing by mobile robots implemented semi-offline path planning. Online path planning was used to generate a collision-free path for each mobile robot from the docking station position to the printing location. When the mobile robots reach the printing positions, the online path planning stops, and the robotic arms mounted on the mobile platforms perform the printing process. The path planning of the printing process itself was

performed offline, and no real-time collision checking and replanning was implemented after each mobile robot moved to the printing location. Elagandula et al. [36] also developed several online and offline algorithms for cooperative printing for mobile robots. In this case, each object to be printed is divided into multiple chunks, each one corresponding to a collection of printing toolpaths. These chunks are generated and assigned to the different robots offline. Similarly to [9], the purpose of the online path planning was to generate a path for each robot to move to a different location for printing. In these studies, the parts to be printed are divided into multiple regions. Each region is assigned to only one robot, reducing potential interactions and cooperation between the robots. Furthermore, the robotic systems performing the actual 3D printing tasks are of high precision with minimum disturbances. This approach shares the same limitations of fully offline path planning, i.e. no account of deviations during printing, uncapable of changing the toolpaths in case of disturbances, and fault intolerance [1,11].

For systems where the printing task is executed by highly dynamic, less precise mechanisms in unpredictable settings, such as cooperative mobile robots, neither of the previous approaches will work. Altering predefined toolpaths is not a feature of offline planning. As shown in this paper, offline algorithms like those suggested in the literature result in collisions when applied to cooperative printing by mobile robots.

The third approach for path planning of robotic systems is online path planning in which the toolpath allocation occurs concurrently with the printing process based on each robot's current state [37]. Thus, in online path planning any deviation or motion disturbance of the mobile robot is considered during the planning process. Furthermore, online path planning solves the limitations of the other approaches, being fault tolerant and the capable of changing toolpaths in case of unexpected deviations or disturbances. However, no work in the literature have explored online path planning for cooperative printing systems, which is the purpose of this work.

In offline path planning, delay-based algorithms are common for collision avoidance. In these algorithms, a delay is added to the travelling printhead motion allowing it to stop at its location until the other printhead moves from the collision location [18]. This was the collision avoidance of many applications. Shen et al. [28] in their cooperative printing by robotic arms divided each printing layer into different regions. When a robot is assigned a region, no other robot is allowed to work on any adjacent region to avoid potential edge collisions. To force this constraint, a delay is added such that the robot stops until the robot occupying the close region is finished. Other researchers also implemented similar techniques for various systems, both physically [12,19,38] and virtually [20,36]. Another implementation for delay-based algorithms is to alter the speed of the travelling printhead instead of fully stopping it. This was implemented by Zhang and Khoshnevis [22] in their cooperative printing for constructions applications. Another method used for collision avoidance in offline path planning, is to replan the layer completely until all the allocations generate collision-free paths [21]. This method is applied when every potential subsequent path for the robots leads to a collision, the planner has the flexibility to reevaluate and modify the previous allocations that caused this situation. This enables the avoidance of an otherwise inevitable collision scenario [20].

These collision avoidance algorithms cannot be applied to online applications for several reasons. Delay-based methods do not always generate a collision-free path as the other printhead might move toward the stationary position of the travelling printhead. Such issue is usually solved in offline planning by returning to a previous collision-free allocation and replan from there [22]. However, online planners cannot replan a previously executed path. Thus, if the mobile robots reached a position in which Delay-based methods cannot solve, the robots must stop, otherwise, they will collide. Hence, online algorithms that guarantee collision avoidance in all scenarios are required. This work proposes a new method that allows mobile robots to escape any problematic path in the layer by sending the travelling mobile robot to home position. In contrast to Delay-based algorithms, the proposed algorithm works in all scenarios and does not require replanning, which represents a key advantage as replanning is not possible in online applications.

Contrary to gantry and robotic arms, mobile robots are not designed with accuracy and precision in mind. The main purpose of using mobile robots is mobility, thus, adopting mobile robots for AM requires further consideration in terms of accuracy. As accuracy is poor in mobile robots, multiple works utilise them in AM to transport and move robotic arms within the manufacturing area [9,39]. To address this issue, this paper presents a mobile robot capable of achieving accurate motion for filament-based material extrusion of plastics without the need of mounting an accurate robotic arm. The effect of the accuracy of the mobile robot is visible on the quality of the parts cooperative printed in this work.

A common assumption in the cooperative printing literature is ignoring the effect of acceleration, i.e.

assuming constant speed [20,35]. This assumption can be justified by using systems with low speeds and very high accelerations. However, this is not always attainable and results in a very low ceiling of printing speeds. Moreover, ignoring the acceleration is not feasible for mobile robots as the accelerations are not as high as those in gantry robots due to factors such as slippage [40]. Therefore, the effect of acceleration is considered in this paper and a linearisation method that accounts for acceleration during the mobile robots' motions is proposed.

This work is motivated by the following gaps in the literature of cooperative path planning:

- Current path planning approaches are not applicable to dynamic and unpredictable systems such as mobile robots.
- Existing collision avoidance techniques are developed for offline purposes and rely on the ability to replan from a previous allocation which is not possible for online applications.
- Common assumptions used in the literature are not applicable to online applications. For example, acceleration cannot be neglected and must be accounted for during collision checking and toolpath allocation.

To the best of the author's knowledge, this is the first work that explores full online path planning for cooperative printing. The paper provides an overview of the developed mobile robot and describes several novel path planning algorithms used in the system. These include the motion model, toolpath allocation, collision checking, and collision avoidance algorithms. In this paper, we develop a novel algorithm for each component to allow full online path planning. We also propose the homing algorithm, a novel online algorithm that guarantees collision avoidance for the cooperative system in all scenarios. It is shown that this algorithm outperforms offline algorithms in avoiding potential collision states. Furthermore, as collision checking is an expensive step in path planning, we propose a fast collision checking algorithm that considers the effect of acceleration of the mobile robots and allows updating the positions in real time. Additionally, the performance of the proposed system is assessed virtually and physically by printing several one-layer and multi-layer objects, which confirm significant improvements in terms of processing speed compared to offline algorithms. Moreover, results proved that the offline systems presented in the literature are insufficient to prevent collisions. In contrast, our proposed system allows two unpredictable systems to print cooperatively without any mid-print failures or collisions.

3. Mobile 3D printer

Two identical custom-made mobile 3D printers were used in this work. The size of the mobile platform is 220 mm × 220 mm and 180 mm in height. Figure 2(a) shows the various components used in the mobile robot and Figure 2(b) shows the mobile robots arranged in the multi-arm configuration [1], with the lower printhead at the bottom, and the upper printhead at the top of the figure. This section details the mobile 3D printer components, hardware, and communication.

3.1. Motion and 3D printing components

The mobile robot motion system consists of four mecanum wheels, goBILDA 96 mm omnidirectional mecanum wheels with 10 rollers (goBILDA, U.S.A.), each connected to a separate motor. Higher number of rollers is chosen to reduce the magnitude of the vertical vibrations due discontinuity between the rollers. To achieve omnidirectionality, two of the wheels are mirror images. The motors connected to the wheels are Dynamixel XH430-W350 DC motors (Dynamixel, South Korea). These motors have internal encoders and are integrated with an internal PID controller, also, the motor specification include: 82 g weight, 3.4 Nm stall torque and 4096 pulse/rev resolution. The motors can carry the load of the platform (4.8 kg) with minimum effort. The Z-axis of the printhead is constructed by an 8 mm threaded rod and two 8 mm linear rods (Misumi, U.S.A.). Another motor is used to push the polymeric filament to the printhead. The motors for both the Z-axis and E-axis are similar to those in the wheels, allowing homogenous and synchronised motion control.

The 3D printing technology utilised in the mobile robots is material extrusion of plastic filaments. Thus, two additional components were required: the printhead and the extruder. The printhead contains a hotend and a cooling fan. The hotend is E3D V6 (E3D, U.K.) with a maximum temperature of 300°C, which enables to print a wide range of materials. Common nozzle sizes used in plastic 3D printing with gantry systems ranges around 0.4 mm [41]. The 0.4 mm nozzle size is out of the range of mobile robots accuracy. However, our mobile robot, which is shown to have higher accuracy than other mobile 3D printers reported in the literature, have around 0.37 mm average error and 0.93. mm maximum error. Thus, the nozzle was chosen to have around double the maximum error for good quality printing and to account for the noisy motion of the mobile robot and its motion resolution. The chosen nozzle is 1.8 mm Bondtech CHT Coated Brass Nozzle (Bondtech, Sweden). 3D printing with mobile

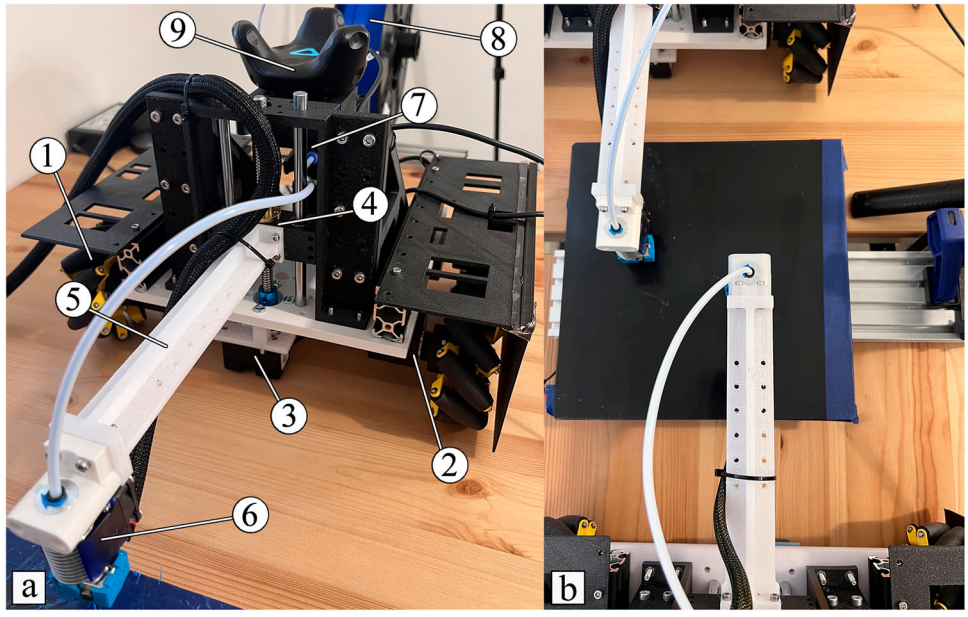

Figure 2. The proposed system. (a) One of the developed mobile robots, showing its main components: (1) mecanum wheel, (2) wheel motor, (3) Z-axis motor, (4) Z-axis structure, (5) printing arm, (6) printhead, (7) extruder, (8) filament spool, and (9) Vive tracker. (b) The two mobile robots arranged in the multi-arm configuration.

robots is not aimed to compete with conventional high accuracy gantry 3D printers that specialise in creating detailed structure, rather to explore other areas such as portability for AM and large-scale 3D printing. For mobile 3D printing in which the accuracy of the mobile robot is in the centimetre range and higher than the nozzle size, the current approach is to mount an accurate multi-axis robotic arm that handles the printing process [12,15,16], however, this further increases the load of the mobile robot and the complexity of the system. For the extruder, a motor is connected to two geared pulleys that push the filament to the printhead. Both mobile robots deposit the molten materials into a heated bed platform that can reach a temperature of 120 °C. The temperature process control is performed by MKS Gen 1.4 electronic board (Makerbase, China), which is operated by Marlin Firmware.

The 3D printing process is carried by an extended one-axis arm. This arm performs the Z-axis motion while the base of the mobile robot performs the planar motion. This makes the mobile 3D printer implements the Multi-arm configuration for cooperative printing [1].

3.2. Localisation and control modules

Accurate localisation of mobile robots is difficult, and they are usually not designed for accurate motion but for mobility. However, 3D printing of plastics requires precise and accurate motion as there are many repeated lines, and any deviation from the generated printing path negatively affects the quality of the printed part and potentially its mechanical properties.

The developed mobile robot uses three different sensors to estimate its position, allowing for accurate and reliable localisation. These sensors are the wheels encoders within each motor, an inertial measurement unit (IMU), and two light-emitting beacons with receivers. The first two sensors are used as local sensors, the encoders measure the rotation velocity of the wheels while the IMU, LPMS-USBAL2 (LP-RESEARCH, Japan), measures the linear acceleration and angular velocity of the mobile robot. However, local sensors are affected by motion disturbances such as skidding and slipping of the wheels [42], thus, a global sensor that is not affected by these disturbances is required. The global sensor used for the developed mobile robot is the HTC Vive (HTC, Taiwan), which is repurposed from its original use in Virtual Reality (VR) applications, where it excelled in providing precise positioning.

The data from the three sensors are combined with the kinematic model of the mobile robot in a linear Kalman filter (KF) [43] to estimate the position and the velocity of the mobile robot. The prediction step of KF uses the data from the IMU and the kinematic model, while the correction step uses the data from both the Vive tracker and the encoders.

The control module developed for the mobile robot is shown in Figure 3. The workflow for the control module is as follows. First, the desired state of the mobile robot (q_d, $\dot{q}_d$) is compared to the estimated state (q, $\dot{q}$) to

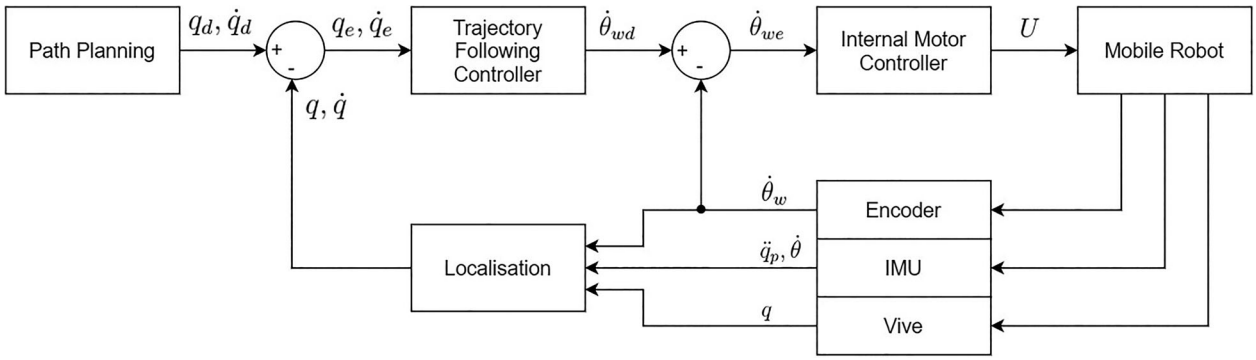

Figure 3. The control block diagram used in the mobile robot, showing the main outer and inner loops with the various sensors used.

find the error state (q_e, $\dot{q}_e$) that must be corrected to ensure the robot is on the planned track. Then, a PID controller takes this state and produces the wheels velocities ($\dot{\theta}_{wd}$) that is needed to achieve the required motion. The internal controllers inside each wheel motor compare the required wheel velocity to the current wheel velocity ($\dot{\theta}_w$) and send the command U that moves the mobile robot and keeps it on track. Finally, each sensor sends its data to the localisation module which closes the control loop by producing the position and velocity estimate. The trajectory following controller is performed 500 times per second while the internal controller inside each motor is performed 8000 times per second.

Using the developed localisation and control modules [44], the mobile robot can perform submillimetre motion accurately, allowing the mobile robot to execute material extrusion of plastic filaments with minimum deviations.

3.3. Communication module

The communication system responsible for sending, receiving, and processing data in the mobile 3D printer was developed on Robot Operation System (ROS) using both Python and C++ programming languages. Python was used for visualisation and non-critical processes (e.g. data recording for future analysis) while C++ was used for modules requiring faster processing times such as the localisation manager and the PID controller. The control of both the heating and cooling elements (such as the hotend and the fans) was based on Marlin Firmware, which is an open-source software used to handle the processing of data in conventional gantry 3D printers. The Marlin Firmware was implemented as a standalone node in ROS to be controlled using the same workflow of the other modules in the mobile robot. Other custom modules developed on ROS include the sensors communicator, motion and motors controllers, data recorder, and the G-code parser. Moreover, as the two mobile robots work cooperatively, a custom node was developed to immediately stop the mobile robots if the distance between them is lower than a safety distance.

The information flow using the various software and modules is shown in Figure 4. The G-code source at the start of the flow is an internal text file if only one of the robots is used as a single-printhead system. However, if the system uses both mobile robots, then the G-code source is taken from the online path planner, which is discussed in the next section.

4. Online path planning

Path planning for cooperative printing comprises toolpath allocation, collision checking, and collision avoidance. This section discusses the proposed algorithms, starting with modelling the mobile robots' motion so the path planning deals with the model instead of the motion of the actual robots for fast online calculations. Finally, the section shows how the mobile robots connect and communicate with the path planner.

Figure 5 provides the overall flowchart summarising the online path planning algorithms discussed in this section.

4.1. Motion model

The path planning of mobile robots can be affected by different factors including the velocity profile, lag in the controller, noise in the measurements, and unpredictability of the motors and the wheels behaviours such as friction and ground unevenness. In this work, the motion model was determined by first analysing the behaviour of the actual mobile robots when performing various 3D printing motions, such as moving with and without printing for long and short lines. This

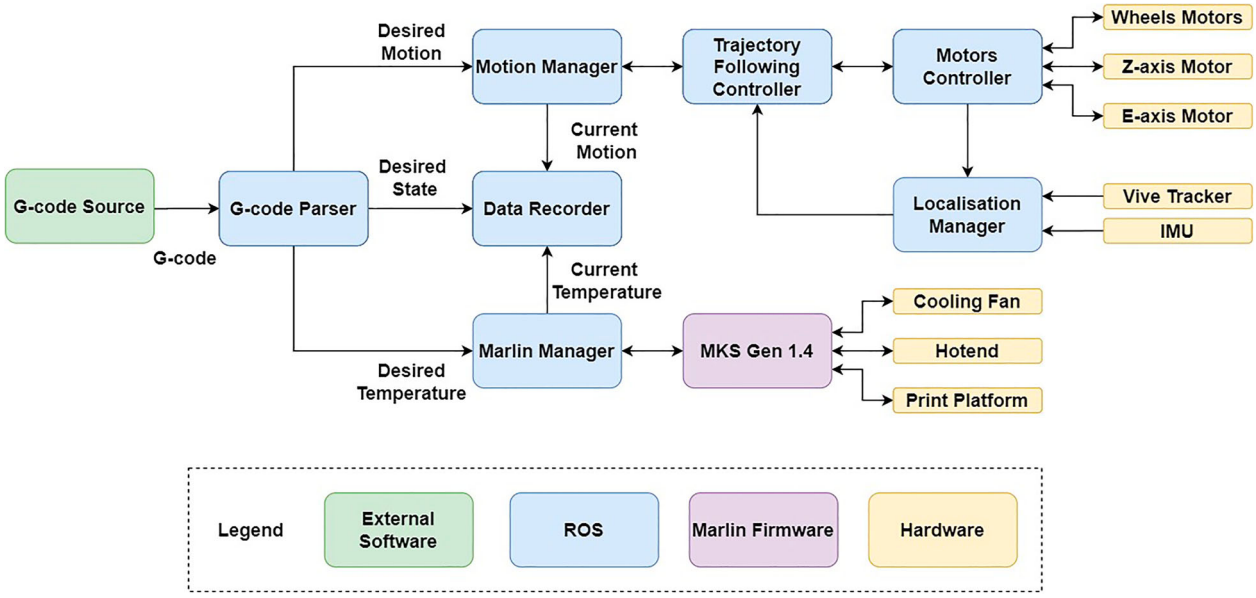

Figure 4. The information flow and the communication system within a single mobile robot.

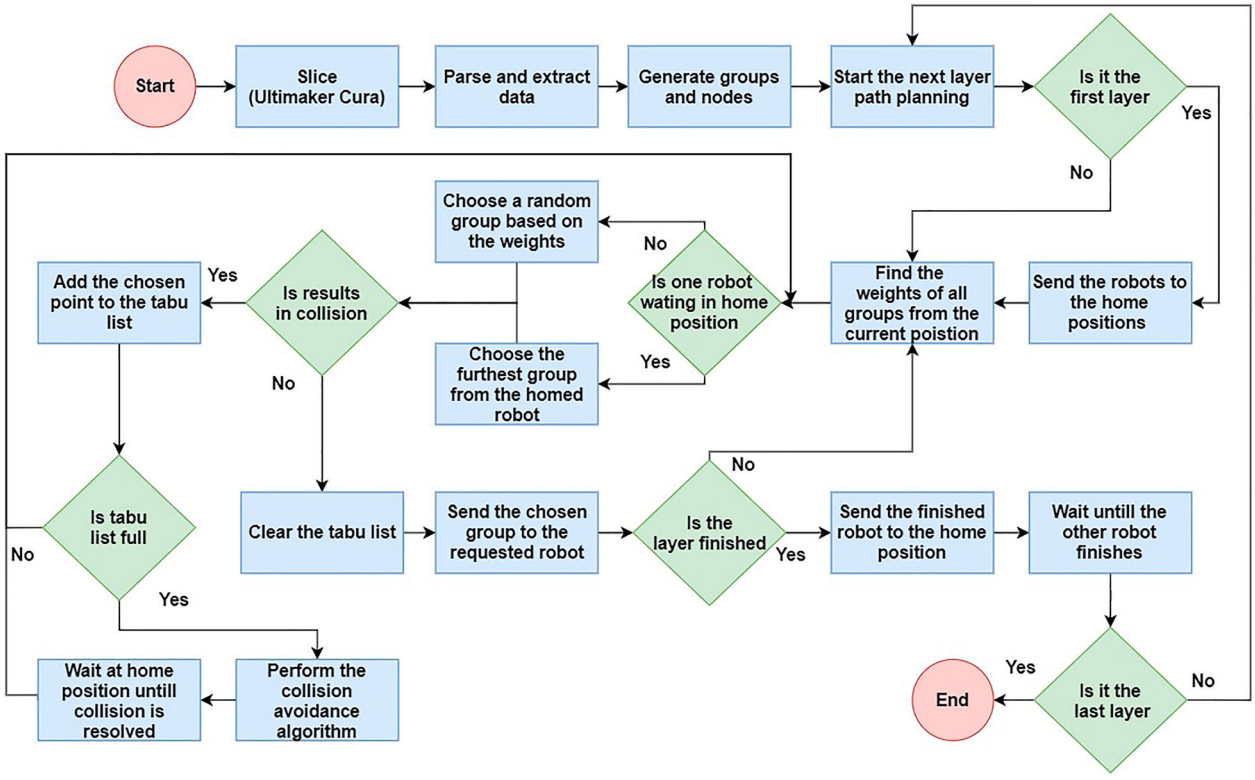

Figure 5. A visual summary of the path planning algorithms developed for the cooperative mobile robots.

behaviour along with the actual commands given to the system were then replicated in the software and compared to a generate a model mimicking the mobile robot motion during the printing process.

The motion of the actual mobile robots resembles a trapezoidal motion profile by design [45]. Thus, our model base structure is a trapezoidal motion profile with additional features. First, the model motion profile has higher coast speed to account for the lower acceleration of the actual mobile robots compared to the requested motion profile acceleration. Moreover, the end of the deceleration phase was delayed by 0.2 s

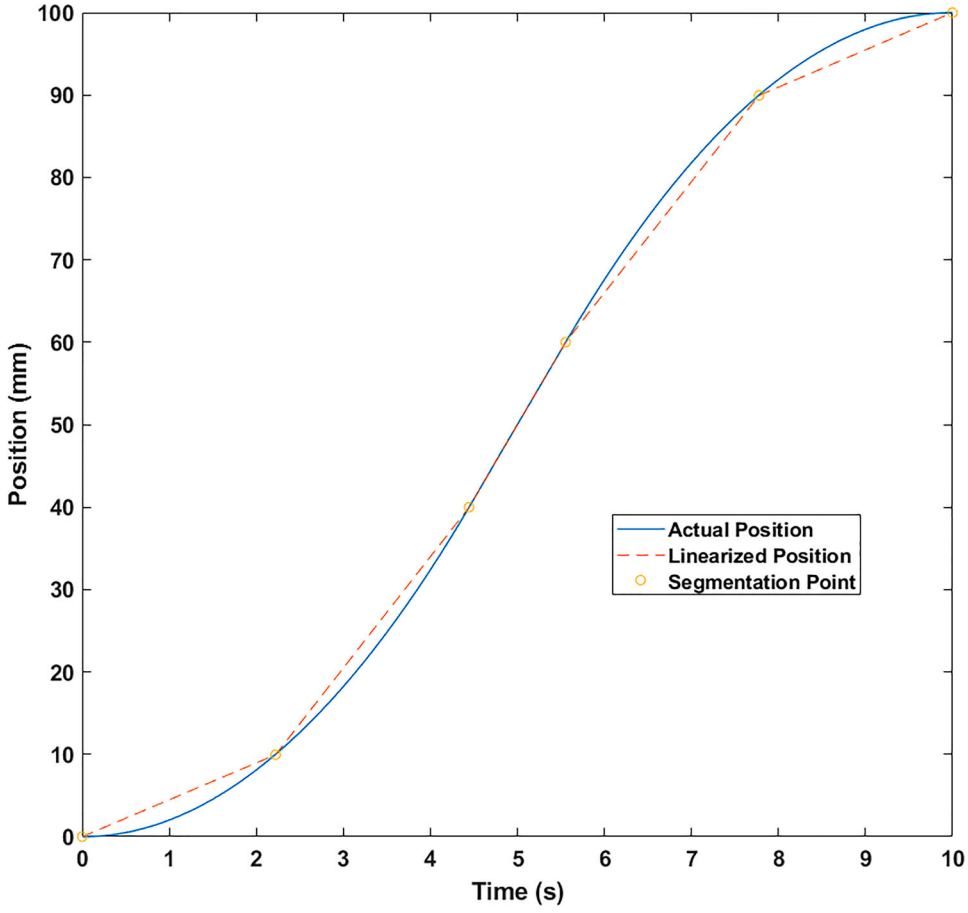

Figure 6. The effect of linearisation model. The actual position of the mobile robot is represented as lines instead of curves.

as the analysed motion of the mobile robots shows a similar delay at the end of the motion.

The position generated from the trapezoidal motion profile is nonlinear, being more costly in terms of computation resources requirements. Analysing the trapezoidal motion, the nonlinearity of the position only occurs during the acceleration and deceleration phases, while the constant velocity phase results in a linear position. We overcome the nonlinearity by linearising the position at the start and the end of the motion. To linearise the quadratic position, we segmented the position into several linear segments, L_s. A large number of segments better resembles the position but increases the collision checking resources consumption. Fewer segments are favoured for fast collision checking, which is preferable for real-time collision checking. Figure 6 shows the actual and linearised positions using $L_s = 2$.

4.2. Toolpath allocation

The toolpath allocation algorithm implemented in this research is based on two algorithms: a local heuristic that accounts for the distance from each mobile robot to the to-be assigned print line, and a tabu search [46] that prevents the toolpath allocation from being stuck in a problematic allocation.

First, the toolpath allocation algorithm receives all the print lines data from a custom G-code parser that reads all the data in a G-code file generated by Cura (Ultimaker, Netherlands), a single-printhead slicing software. These print lines are grouped based on the distance between them into many groups. The grouping algorithm begins by forming an empty group, then selects the first line, adding adjacent lines until a defined length threshold is met. Then, a new group is created for the next set of lines, and the process is repeated until all print lines of the layer are assigned to their respective groups. Then, the algorithm work on allocating groups of lines to the robots instead of allocating individual lines, thus reducing the frequency of allocation. However, the threshold must not be overly lengthy, as the expected planned path may not align with the mobile robot's actual path, given that the error between the planned and actual paths can diverge over time due to model inaccuracies and

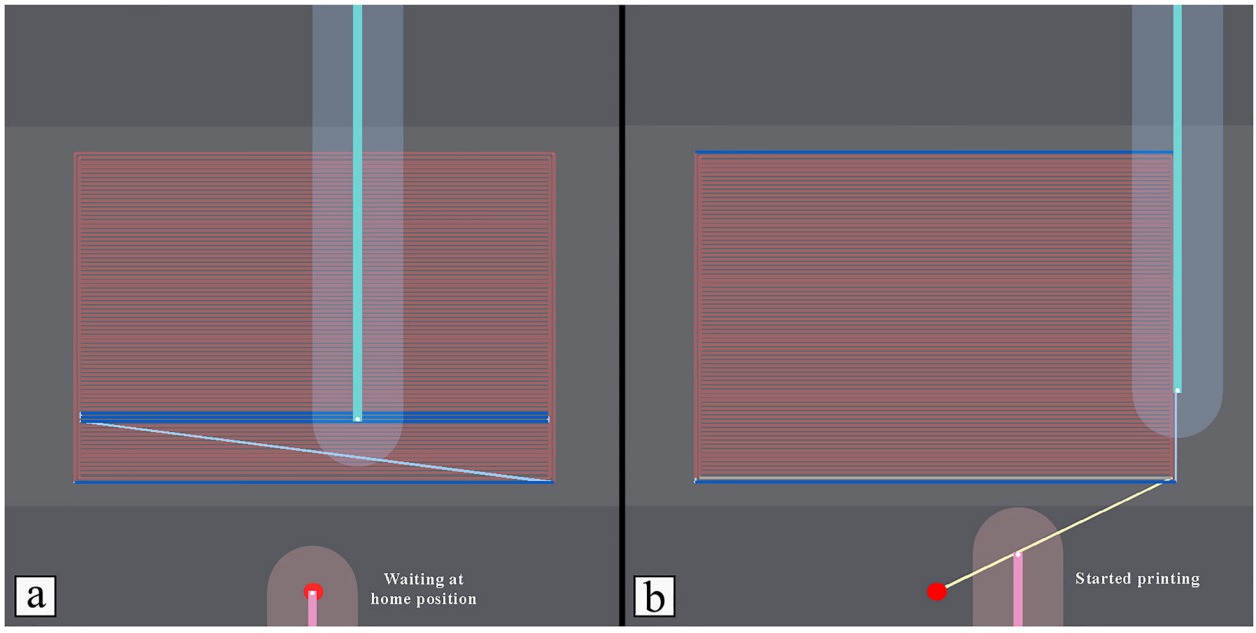

Figure 7. The effect of changing the allocation algorithm when one printhead is homed: (a) allocation based on Equation (1) leads to the lower printhead waiting at the home position; (b) allocation based on the farthest point from the homed printhead allowed the printhead to have a toolpath allocated.

latency. Therefore, this value should accommodate the system's overtime deviation. Conversely, a value too small might cause each group to consist of merely a single print line, undermining the utility of the grouping algorithm.

Before the allocation, the software sends all the robots to the corresponding predefined home positions, starting the allocating only when all the robots confirm their arrival at the home position. The last mobile robot that reaches the home position is the first which requests a print line to be allocated to it.

The toolpath allocation is performed on request from the mobile robots, by sending to the toolpath allocator its current real-time position and whether it is the upper or the lower printhead. Based on this information, the toolpath allocator performs a local heuristic that assigns a weight w_{ci} to all the groups of print lines, choosing a random one based on these weights. The weight of the ith group w_{ci} is given by:

$$w_{ci} = d_{oi}^{a}\, d_{mi}^{b} \tag{1}$$

where d_{mi} and d_{oi} are the distances from the starting of the ith group to the calling and the other printheads, respectively. The exponents a and b play a crucial role in determining the significance of choosing nearby or far-off groups for the calling and the other printhead, respectively. The exponents' values were determined through a series of controlled experiments, where the mobile robots were tasked with air-printing a 120 mm square layer multiple times. For each trial, the exponents were systematically adjusted. The experiment that consistently yielded the shortest printing time had values of $a = 2$ and $b = -4$.

The group selected from this heuristic is sent to the collision checker for validation. In the absence of any collisions, the algorithm proceeds to the next set of groups. However, if a collision is detected, the group responsible for it is marked and added to a tabu list, and an alternative group is chosen for the same printhead. The tabu list serves to prevent the algorithm from selecting the same group again, which can result in an infinite loop if its w_{ci} value is significantly higher than that of other groups. The tabu list is reset after each successful group allocation to ensure that the algorithm can consider all groups in the subsequent iterations.

If the toolpath allocation generates potential collisions, these must be avoided by the collision avoidance algorithm (see Section 4.4) and in these cases, the robot is repositioned to its home position. However, this represents a period of inactivity, which impacts the efficiency of cooperative tasks. To minimise these idle periods, the toolpath allocator follows a strategic approach when assigning toolpaths. Rather than providing a path to the nonhomed printhead following Equation (1), it allocates the path based on the furthest point from the home position of the homed printhead. This enables the homed printhead to find a path while homed. This strategic allocation (Figure 7) is only implemented a finite number of times before reverting to considering points nearest to the home, thereby

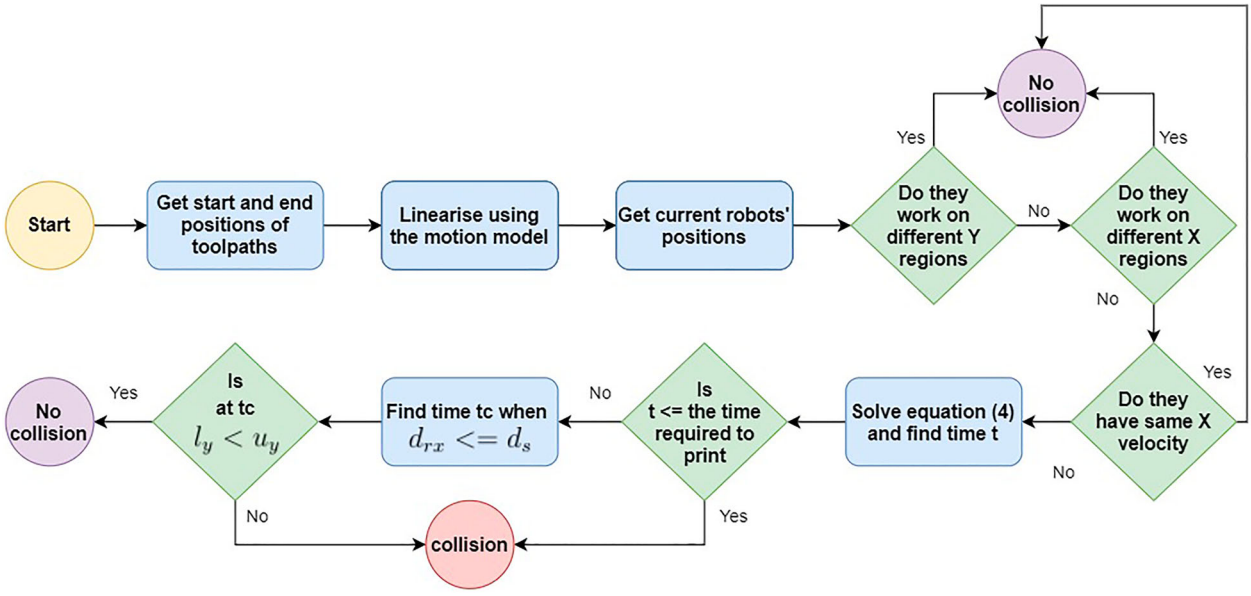

Figure 8. Flowchart of the collision checking algorithm.

preventing an excessive number of extended travel motions.

Figure 7(a) depicts a situation where the toolpath allocator has assigned a path to the upper printhead based on Equation (1). Consequently, the lower printhead remains at the home position due to the absence of a collision-free toolpath. On the other hand, in Figure 7(b), the toolpath allocator designates the farthest toolpath from the home position for the lower printhead to the upper printhead. This strategic move allows the lower printhead to obtain a collision-free toolpath, thus enhancing the overall operational efficiency.

Before the mobile robot finishes printing its allocated group of print lines, it requests from the toolpath allocator another set of toolpaths and sends its current position. The time before requesting the next toolpath is based on the processing speed of the toolpath allocator. Using the motion model and the current position of both robots, the toolpath allocator predicts the position of both mobile robots when the requesting robot arrives at the end of the current printing group and allocates a new group of print lines to the requesting robot. Thus, faster processing speed with fast algorithms reduces the time between requesting new toolpaths and arriving at the destination, which results in more accurate predictions.

If all the toolpaths of the current layer are already allocated and printed, the robot is sent to the home position until the other robot finishes its toolpaths. When all robots complete their tasks, the toolpath allocator moves to the next layer. This process continues until the object is fully printed.

4.3. Collision checking

The collision checker (Figure 8) accepts as input the start and end positions of the toolpath that is assigned to each mobile robot. The algorithm uses the developed motion model of the mobile robot (see Section 4.1) to predict the chances of collisions between two mobile robots. As the position between the start and end points is nonlinear, it uses the linearisation method and checks for collisions assuming a constant speed. As errors can be accumulated, the motion model cannot be used for a long sequence of motions without correcting the positions of mobile robots in real time. Thus, the collision checker only analyses a small set of future lines before stopping until the mobile robots reach the end of the analysed lines.

As it is rare that two line segments assigned to the mobile robots have the same length, one mobile robot usually finishes earlier than the other. In this case, the collision checker remembers where the late mobile robot stopped and what remained for it. However, the stopped position is updated and corrected with the actual position of the mobile robot in real time. This is acquired by allowing the collision checker to subscribe to the module responsible for the localisation of the mobile robots. The next input to the collision checker is the new motion of the just-finished mobile robot. Therefore, since the current and remaining positions of the late mobile robot are already determined, no linearisation is performed on it.

The collision checking algorithm checks if an intersection is possible between any two toolpaths. Since the

motion is linearised they are simply line segments, making the checking algorithm straightforward and very fast. Assuming that the lower printhead is moving from l_s to l_f with a velocity v_l and that the upper printhead is moving from u_s to u_f with a velocity v_u, collision occurs if the distance between them is lower than a safety distance d_s:

$$(l_x - u_x)^2 + (l_y - u_y)^2 \leq d_s^2 \tag{2}$$

Let d_r represent the relative displacement vector between the two printheads, defined as $d_r = l - u$ at time t, d_{rs} denote the starting relative displacement vector, and v_r represent the relative velocity vector $v_r = v_l - v_u$. The equation for the relative motion between the two printheads becomes:

$$d_r = d_{rs} + v_r t \tag{3}$$

By transforming Equation (2) to account for relative displacements and subsequently combining it with Equation (3), the following quadratic equation is obtained:

$$v_r^T v_r \, t^2 + 2 d_{rs}^T v_r t + d_{rs}^T d_{rs} - d_s^2 \leq 0 \tag{4}$$

A collision between the printheads will occur if the solution of Equation (4) yields a real number. This condition is satisfied when:

$$(2 d_{rs}^T v_r)^2 - (4 v_r^T v_r)(d_{rs}^T d_{rs} - d_s) \geq 0 \tag{5}$$

However, these equations conceptualise the print line segment as an infinite line. As a result, the time calculated from Equation (5) needs to be compared with the time necessary to complete the current line segment. If the former is greater, then the toolpaths are collision-free.

The previous operations only consider the printheads, not the link carrying them, which might collide in any planar direction. Collisions in the Y-axis are always started by the printheads, thus, it is not needed to check for arm collisions in the Y-axis as the previous step already considers them.

The X-axis collisions between the arms are checked by computing the time t_c at which the relative distance in the X-axis between the mobile robots reaches the safety distance, i.e. when $d_{rx} \leq d_s$. At this instant of time, there are two possible scenarios:

1. No collisions: if the lower printhead is moving beneath the upper printhead, i.e. $l_y < u_y$.
2. Collisions: if the lower printhead is moving above the upper printhead, i.e. $l_y > u_y$.

The third scenario ($l_y = u_y$) already ruled out by the operations performed using Equation (5), note that this is also true if the printheads are closer than the safety distance in any direction.

To further make the collision checker faster, several cases requiring minimum computation resources are checked first. One case is that if both printheads work in different regions, which is true if the following condition is satisfied:

$$\begin{aligned} &\min(u_{sy}, u_{fy}) > \max(l_{sy}, l_{fy}) \;\; and \\ &\min(u_{sy}, u_{fy}) - \max(l_{sy}, l_{fy}) \geq d_s \end{aligned} \tag{6}$$

These equations are used to verify that the mobile robots work on different regions based on the Y positions, a similar evaluation is performed for the X positions. Another case is if both printheads have the same velocity direction and magnitude in the X direction.

4.4. Collision avoidance

The collision avoidance algorithm developed in this work is based on sending one of the mobile robots to the home position to escape the collision situation. The home position of the lower printhead is the middle point in the lower side of a rectangle that bounds the layer, and for the upper printhead it is the middle point of the upper side of the rectangle. Figure 9 illustrates the developed algorithm. In this figure, two printheads are working on printing a rounded square and the lower printhead (pink) is about to collide with the upper printhead (blue).

The method of collision avoidance is performed in two steps: setup and sequence generation. The setup (Figure 9(a)) is performed at the start of the layer for each printhead. It starts by generating a bounding rectangle R of the part and offsets it by the safety distance, the bottom left corner S_{bl} and the top right corner S_{tr} of the safety bounding rectangle S are given by:

$$\begin{aligned} S_{bl} &= (R_{blx} - d_s, R_{bly} - d_s) \\ S_{tr} &= (R_{trx} + d_s, R_{try} + d_s) \end{aligned} \tag{7}$$

The algorithm then allocates the middle point of the side nearest to the base as the home position. The lower and upper home positions P_{hl} and P_{hu} are:

$$\begin{aligned} P_{hl} &= \left(\frac{S_{blx} + S_{trx}}{2}, S_{bly} \right) \\ P_{hu} &= \left(\frac{S_{blx} + S_{trx}}{2}, S_{try} \right) \end{aligned} \tag{8}$$

The setup of the homing algorithm is performed only once per layer, as the locations are based on the geometry of the layer. Provided that the layer geometry remains unchanged from the previous layer, the setup is retrieved from that layer instead of being recalculated.

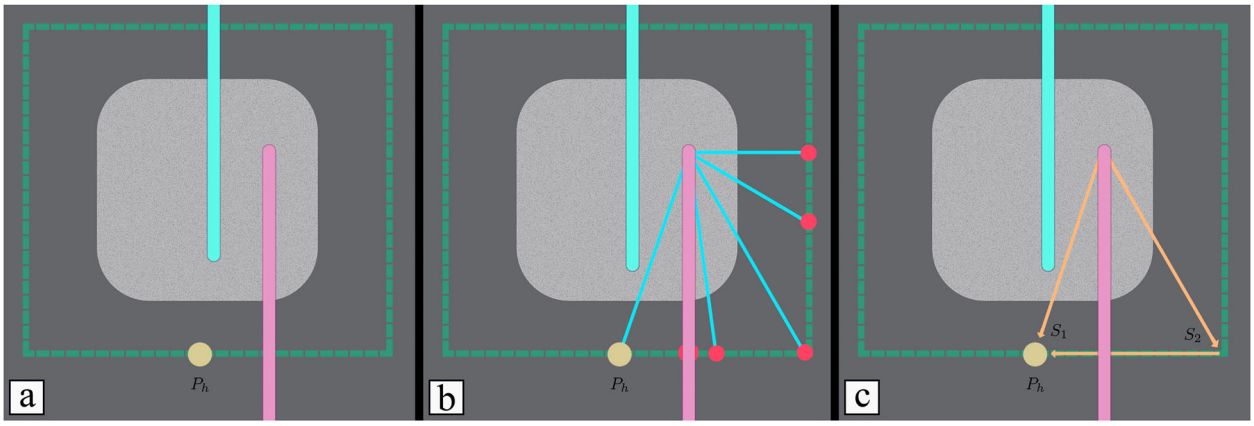

Figure 9. The homing algorithm. (a) Setup and generating the bounding rectangle (green); (b) multiple rays are sent from the position of the printhead to the border of the bounding rectangle; (c) example of two sequences generated from the previous step.

If the object is a 2D layer that has been extruded in 3D, i.e. all layers have identical geometry, this process is performed only once for the entire object.

The second step is the homing sequence generation. Here the algorithm sends several rays from the position of the printhead (Figure 9(b)). Each ray intersects a side of the bounding rectangle at a point P_i, which corresponds to the first point in the homing sequence. If the point shares the same Y coordinate of the home position, i.e. $P_{iy} = P_{hy}$, then the next point in the homing sequence is the home position. The sequence S_i becomes:

$$S_i = [P_i, \quad P_h] \tag{9}$$

However, if the Y coordinate of the point is different from the home position Y coordinate, then a point that shares the same Y coordinate with the home position is added to the sequence before the home position, this point is $P_i = (P_{ix}, \ P_{hy})$. This is required so that the homed printhead should only move on the bounding rectangle. Thus, the sequence in this case is given by:

$$S_i = [P_i, \quad (P_{ix}, P_{hy}), \quad P_h] \tag{10}$$

The final point added to the homing sequence is the home position itself because at this position it has more access to collision-free regions on the layer compared to any other point on the bounding rectangle. Figure 9(c) shows two examples of sequences that are generated from the points shown in Figure 9(b), S_1 contains only one path while S_2 contains two, but S_1 passes close to the other printhead.

Rays sending algorithm is as follows for the lower printhead, the upper printhead is similar but mirrored. Two rays are sent from the position of the printhead P, one to the closest left or right side (R_x) and one to P_h the home position (R_h), with the angles of rays given by:

$$\begin{aligned} R_x &= \begin{cases} 0^\circ & P_x \geq P_{hx} \\ 180^\circ & P_x < P_{hx} \end{cases} \\ R_h &= \tan^{-1}\left(\frac{P_y - P_{hy}}{P_x - P_{hx}}\right) \end{aligned} \tag{11}$$

Then, the algorithm sends several rays between these two rays starting at the ray sent to the home position. Those rays are sent at angle increment until the ray surpasses the horizontal ray. The direction of increment is the one that reaches the horizontal ray in fewer rays. A lower angle increment increases the number of rays and allows more options for sequences but increases the computation time. In our implementation, we used 6 rays. An additional ray is sent vertically and lies within the region. The horizontal and vertical rays are required as one of these rays is always shortest than any other ray and allows the printhead to reach the bounding rectangle faster. Selecting the initial ray as the home position and the final ray as the horizontal one is because any rays beyond them will not provide any shorter or collision-free rays that haven't been previously considered. This is due to the shortest ray to the bounding rectangle, which is the fastest to avoid collisions, is the horizontal one. The ray with the shortest overall sequence distance is the one to the home position. This is implemented as the goal of the algorithm is to send the problematic printhead to its home position to avoid collisions, and directly sending it to the home position might be already collision-free and thus reducing the travelling time.

The homing algorithm generates a list of homing sequences. The number of elements in this list is equal to the number of rays sent in the previous step. The list is ordered based on the total length of the sequence.

The toolpath allocator checks each sequence and stops when a sequence path is collision-free. This check is only performed for the first part of the sequence, i.e. the path to the point that intersects the bounding rectangle, as any motion after that is guaranteed to be collision-free due to the bounding rectangle method.

4.5. Integration of unity and ROS

The path planner algorithms were implemented in Unity (version 2021.3 running on Windows 10) using C# as a programming language, while the mobile robots' software was developed in ROS (version Noetic running on Ubuntu 20.04) using C++ and Python. The path planner is not directly connected to the mobile robots as it is used in a standalone personal computer (3.6 GHz Intel Core i7-9700 K CPU with 16 GB memory) that is powerful enough to run the algorithms in real time without interfering with the processes on the mobile robots. The Unity Robotics package (Unity, U.S.A.), based on Transmission Control Protocol (TCP), is used to connect the mobile robots with the path planner and to allow data transmission and communication between them.

From the Unity and the path planner software side, the ROS TCP Connector is used. This allows the path planner to send messages to the mobile robots and receive data in real time from them. On ROS side, the ROS TCP Endpoint is used as a node, allowing the transmission of messages from the mobile robots to the path planner.

The path planner subscribes to ROS topics that the mobile robots publish their positions to and implements a ROS service that is called by the mobile robots when it requests new toolpaths. The data sent to the mobile robot includes the position of the next point, extruding distance, and the Z height. The extruding distance is zero if the toolpath allocated is a travel motion and the Z height is only changed if the robots finished the current layer.

In any communication between two systems, a certain latency delay is inevitable. High latency can adversely impact motion planning and control. The positioning error e due to latency delay t_l while a mobile robot is moving at speed s can be mathematically represented as $e = t_l s$. In our application, the average round-trip latency – starting from Unity, going to ROS, and then back to Unity – is found to be approximately 8 ms. With 18 mm/s speed used for the experimental works in Section 5.2, the error due to latency is 0.14 mm. This level of error is negligible and does not significantly affect the system's performance. However, for applications with larger errors, one potential mitigation strategy is to increase the safety distance d_s by e. This adjustment allows errors introduced by latency to be considered for path planning, and the collision checker in particular, thereby enhancing the system's overall reliability.

5. Results and discussion

This section first details our work in modelling the developed algorithms in a simulation software. The simulation was used to determine the effect of linearising the motions model, verify the path planning algorithms and compare them to offline path planning. Following this, several experimental works are presented and discussed. Finally, the section concludes by highlighting the observed limitations of the developed system.

5.1. Simulation software

To test the results of the path planning and the accuracy of the generated model and the path planner, a simulation was developed in Unity and C#. In this simulation, the motion of the mobile robots implements a PID controller tuned to behave like the actual robots, and the sensor data is affected by a Gaussian noise that matches the noises of the localisation system. Furthermore, additional disturbances were added randomly throughout the motion of the robot to account for the uncalculated external disturbances. This model is more accurate than the one described in Section 4.1, but much more complex and more computationally heavy when used for real-time path planning. The purpose of the simulation model is to replace the actual mobile robots so that the system can be checked virtually before being implemented physically. The collision checking and avoidance uses the previously discussed linear model. Figure 10 shows the developed simulation software. In this figure, two printheads (upper in blue and lower in red) are printing a single layer cooperatively. The simulator only models the printheads' arms, not the actual mobile robots. However, the motions of the arms are based on a model that mimics the actual motion of the mobile robots. The legend shown in Figure 10 is used to explain all the figures showing the simulator view.

5.1.1. The effect of linearisation

Figure 11 shows the effect of the linearisation performed in section 4.1 on both the maximum error of the motion model and the computation time. In this figure, the maximum error is based on a trapezoidal motion profile of a mobile robot moving 160 mm with 18 mm/s coarse speed and 10.1 mm/s^2 acceleration. As expected, a larger number of segments results in

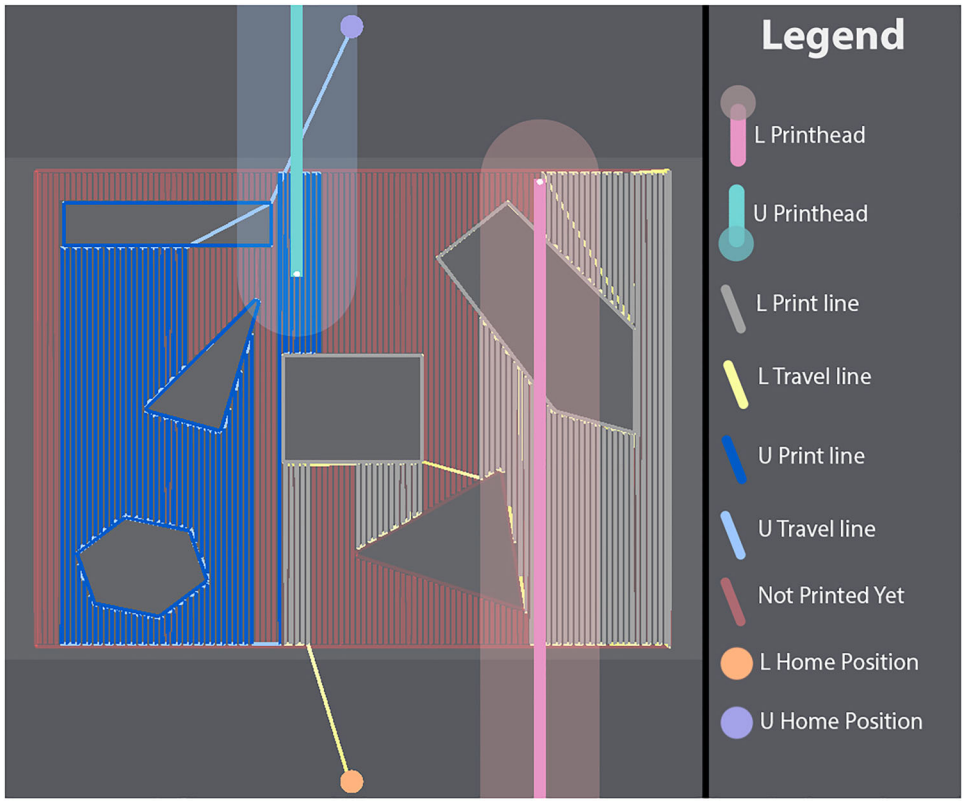

Figure 10. Two printheads printing a sample layer in the developed simulation software. L stands for lower and U for upper. The legend is used in subsequent figures showing the simulation software.

better resembling the motion profile, which reduces the errors. However, computation time also increases with the number of segments. The errors drastically reduced when using segmentation of 3 lines or more. For a large number of segments, the accuracy improvement becomes smaller. For example, the maximum error reduces by only 0.02 mm when changing from 7 segments to 8 segments. However, the computation time increases almost linearly with the number of segments used. In our application, we chose 3 segments as the trade-off between accuracy and computation time is suitable for the system compared to other options. The error at 3 segments is 0.44 mm, which is less than 1% of the size of the printhead. Thus, increasing the number of segments is practically insignificant in terms of accuracy. However, it is important to notice that higher accuracy results in higher computation time, which can be critical in real-time applications.

Furthermore, using a linear model that completely ignores the acceleration, as reported in the literature on offline path planning [22,35], results in large deviations that lead to inevitable collisions. Using the previous parameters, the maximum error when ignoring the acceleration is 11.1 mm with a 7.1 mm average. If no online correction is made, these errors accumulate over time unboundedly. However, online correction is not enough as the errors are very large, making the system unpredictable, and results in the collision checker not functioning correctly.

5.1.2. Cooperative printing simulation

Three different objects were considered in the developed simulation software to test the performance and reliability of the algorithms. Three layers have been simulated: a solid rectangle of size 160 mm × 100 mm (Case 1 Figure 12(a)), a 160 mm × 120 mm rectangle with different holes (Case 2 Figure 12(b)), and a grid of equally spaced 25 mm squares (Case 3 Figure 12(c)). All objects were sliced with 1.4 mm line width in Cura, and the slicer view is shown at the top of Figure 12. Each layer was planned in the simulator 20 times.

The bottom of Figure 12 shows the toolpath allocated by the algorithms in the simulation software. Travel lines out of the layers are of two types: the start and end of the printing process, and the homing algorithm taking place to prevent collisions. The reduction of printing time varies based on the layer geometry. Case 1 showed the lowest (41%) average printing time reduction. However, the variation in printing time reduction is very large as the standard deviation is 3.6. The printing time reduction of Case 1 is strongly related to the first print line allocated to the first

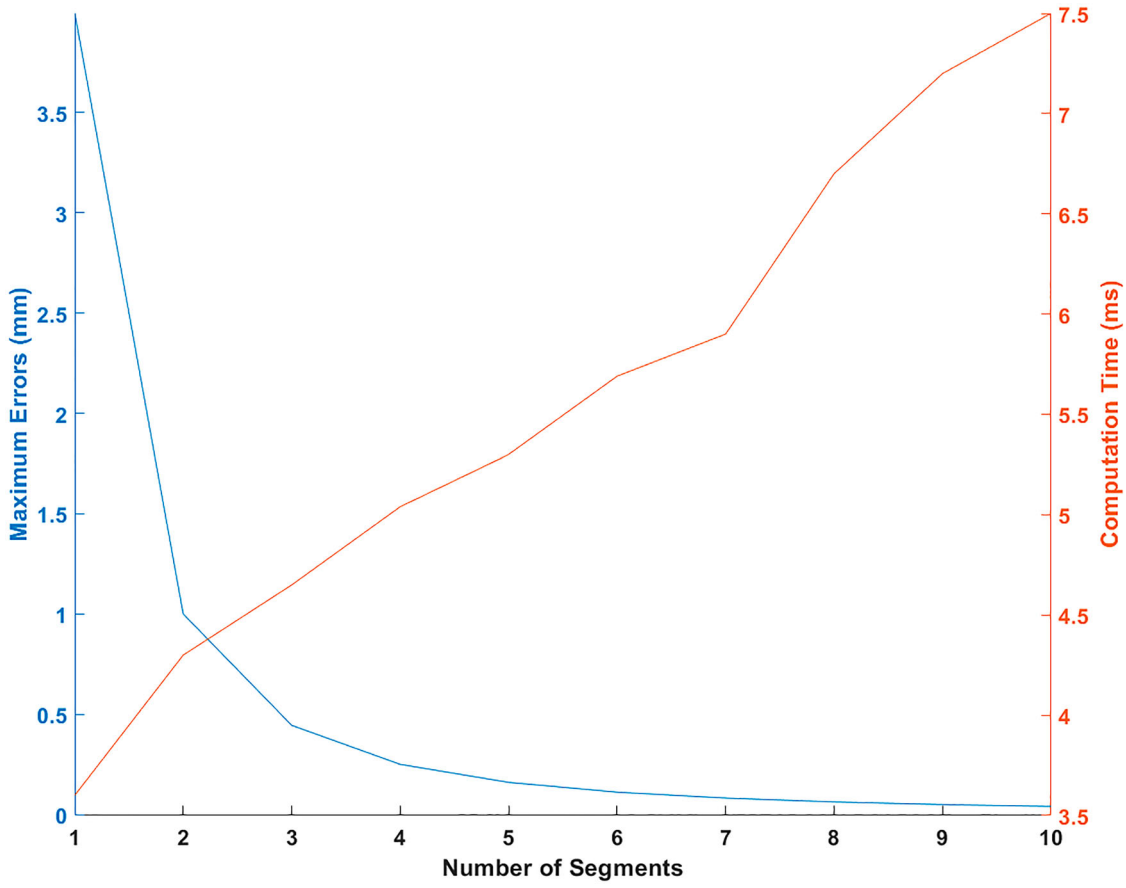

Figure 11. The effect of choosing the number of segmentations on the accuracy and the computation time.

printhead. The printing time reduction is maximum when the first printhead was allocated the middle line and printed to the right wall while the other printhead started at the left wall and printed toward the middle. However, this is not always possible due to the randomness of the allocation. Case 2 showed better results, with an average of 43% of printing time reduction and 2.8 standard deviation. Contrary to Case 1, the starting line allocation was not observed to largely affect the printing time reduction. However, Case 2 exhibited a large number of travel lines with long distances which negatively affected the printing time reduction. Case 3 corresponds to the layer with the highest average printing time reduction reaching 45% and with the lowest variation, 2.1 standard deviation. The chance of collisions found in Case 3 was very low compared to the other layers and thus the homing algorithm was rarely applied, due to the clustering of the squares that makes the printheads spend more time in a narrow region compared to long lines in Cases 1 and 2.

Given that a conventional single-printhead system spends 10.9 min for Case 1, 11.35 min for Case 2, and 7.58 min for Case 3, the cooperative system developed in this work significantly improves efficiency. Specifically, the average time spent by our cooperative system is 6.43 min for Case 1, 6.47 min for Case 2, and 4.17 min for Case 3. Such efficiency gains demonstrate the effectiveness of the cooperative system in reducing printing time across a variety of complex layers, thereby showcasing its potential for broader applications in 3D printing as slow printing speed is one of the major limitations of AM.

The developed simulation was used to compare the performance of both online and offline path planning for cooperative printing. In an offline setting, the path planner is allowed to replan certain paths or even the whole layer if a collision situation cannot be avoided. This offline system is compared to our online system in terms of performance and resources consumed.

Online algorithms cannot guarantee a specific level of performance or a minimum printing time reduction. This is because online path planning algorithms must make decisions based on incomplete information and are not allowed to replan an executed path. As a result, the performance of online algorithms can significantly vary depending on the specific problem, and it can be challenging to predict or specify a minimum level of performance.

Case 2 was also used to test offline path planning. The offline path planner reduced the printing time to 46%

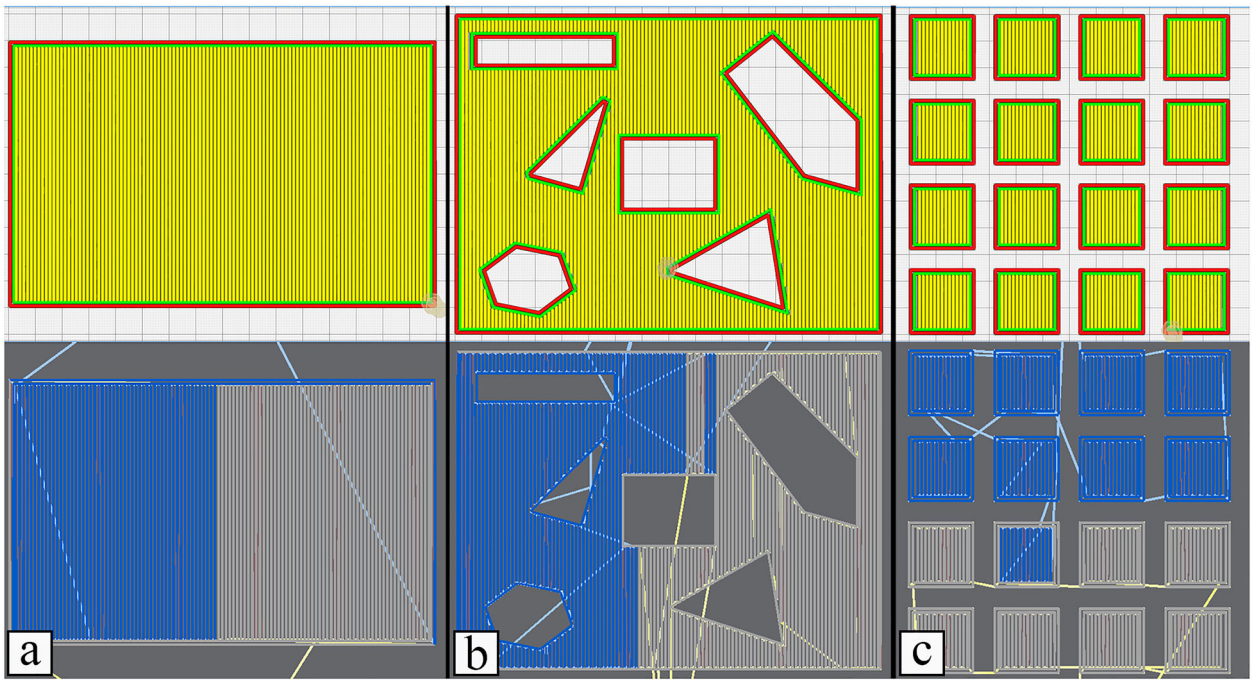

Figure 12. Three layers tested in the simulation software (a) a solid rectangle, (b) a rectangle with multiple holes, (c) a grid of squares. Top is the slicer view (yellow lines for infill and red and green for outer and inner contour, respectively) and bottom is an example of a planned layer.

with a standard deviation of 1.3. These results highlight the limitation of online algorithms in the sense that the printing time increase is unpredictable beforehand and thus cannot be specified to a minimum value. On average, online algorithms produced higher printing times as the collision avoidance is based on the homing algorithm. This is visible when the online path planning plans a layer, and no collision occurs that requires homing. In such cases, the average printing times reached values like the ones produced by the offline algorithms. However, online algorithms are much faster when solving layers with collisions due to the homing algorithm, and no replanning is needed. In this case, the online algorithm required on average half the computation time compared to the offline one.

Next, the path planning is compared to a layer for which the optimal printing time is known in advance. This particular layer (Figure 13(a)) is a simplified version of the one discussed in Case 1, with the outer contour removed to ease calculations. Assuming constant speed, the optimal printing time for this layer occurs when one printhead is responsible for the left region and the other for the right region. The maximum printing time between the two printheads is given by the following equation:

$$\left(d_{hs} + \frac{n}{2}l + \left(\frac{n}{2} - 1\right)w + d_{hl}\right)\frac{1}{s} \tag{12}$$

where d_{hs} and d_{hf} are the distances from the home position to the nearest and farthest points, respectively, n is the number of lines, l is the length of each line, w is the line spacing, and s is the printing speed. Figure 13(b) is a schematic diagram of the layer where the variables are clearly annotated. For this layer, $n = 114$, $l = 98.6$ mm, $w = 1.4$ mm, and the printing speed $s = 18$ mm/s. The distances from the home position to the nearest and farthest points are 36.01 and 156.12 mm for both printheads, respectively. The cooperative system's optimal printing time is 327.26 s, compared to 642.91 s for a conventional single-printhead setup, a reduction of 49.10% in printing time. When the time spent travelling to and from the home positions is disregarded, the time savings increase to 50%, this is more accurate for longer print durations.

Upon simulating the developed online path planner for this specific layer 30 times, a range of printing time reductions is observed, with a minimum of 40.18% and a maximum of 49.05%. The average reduction across all simulations was 47.65%. Figure 13(c,d) shows two examples of printed parts. The first case (Figure 13(c)) approaches the upper bound of printing time reduction while the second case (Figure 13(d)) is near the lower bound. The likelihood of achieving higher printing time reductions increases when the starting printhead is assigned to the middle toolpath. Conversely, when the toolpath allocation becomes asymmetrical, the

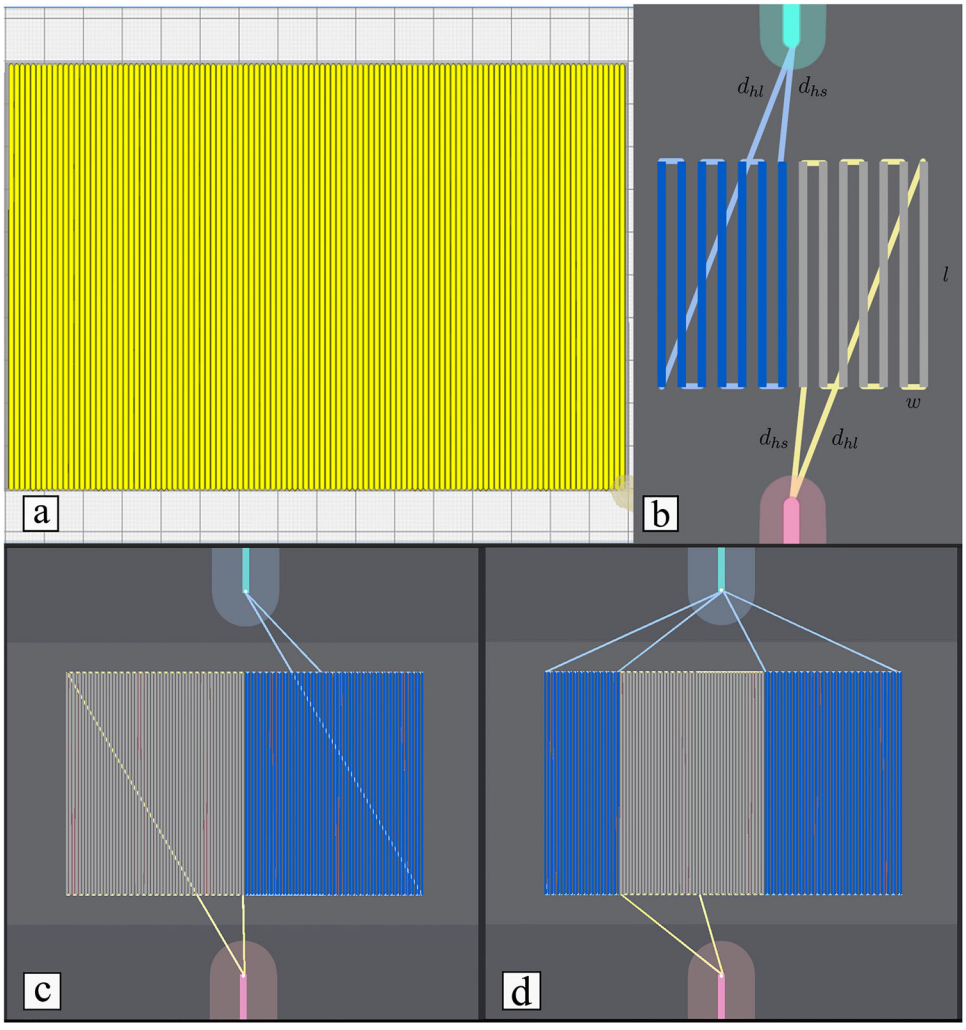

Figure 13. Optimal printing of a layer. (a) A square layer without contour, (b) schematic diagram of the layer in which the two printheads optimally cooperate, (c) a symmetrical and (d) asymmetrical prints.

efficiency diminishes resulting in decreasing the reduction ratio to 40.18%. The primary cause for this decrease appears to be due to less optimal utilisation of each printhead's time and movement. Specifically, such asymmetrical assignments may necessitate additional travel time for each printhead to reach its respective home position or could result in idle periods where one printhead waits for the other to complete its task, thereby reducing the overall system efficiency. The influence of symmetricity is also observed during the physical evaluation of the cooperative printing process in Section 5.2.2.

5.1.3. Collision avoidance evaluation

The homing algorithm's purpose for collision avoidance can be seen in the simulation. Figure 14 is a square-layer version of Case 2. As indicated (Figure 14(a)) the printhead of the lower mobile robot (red) requested a new toolpath to be allocated to it. The remaining toolpaths of this layer are the walls on the right and top. However, both walls if allocated to the lower printhead will result in a collision with the other printhead which is currently moving to the right to print the bottom wall. Thus, the homing algorithm generated a homing sequence for the lower printhead to escape the collision state as shown in Figure 14(b).

To evaluate the collision avoidance algorithm, two additional cases were considered. The first case features a spiral layer with an approximate size of 102 mm × 99 mm. This layer was considered due to its compact size and intricate curvature. Smaller layers compared to the size of the printhead increases the chance of collisions. In standard 3D printing slicing software, curves are broken down into multiple straight lines, thereby increasing the computational load on the path planning algorithm. However, using the grouping preprocessing method, every 25 mm is

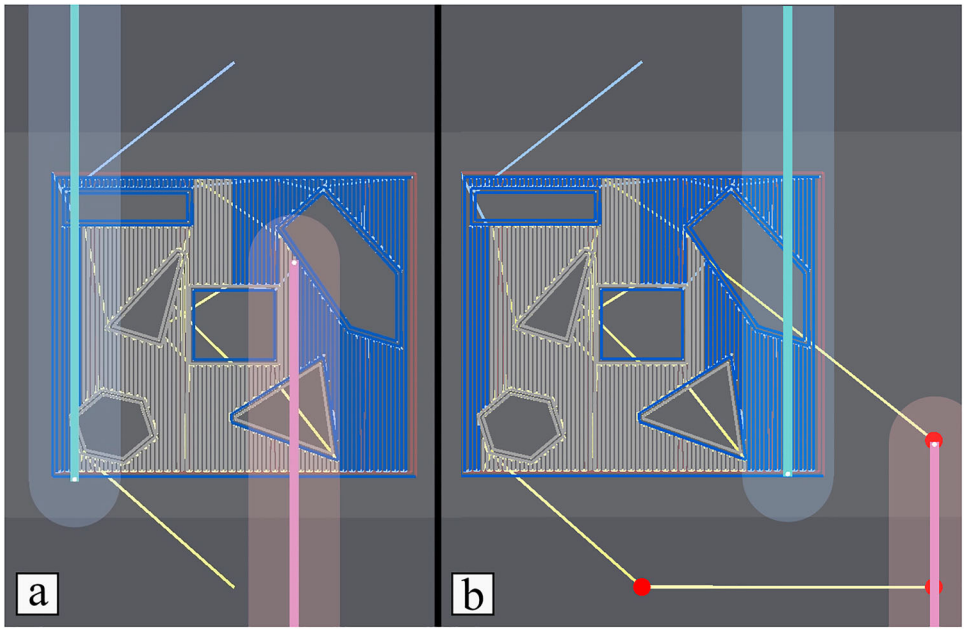

Figure 14. The homing algorithm taking place to prevent collision (a) the lower printhead finished its toolpath and requests a new one; (b) the homing algorithm moves the lower printhead from the way of the upper printhead. Red dots are the points generated by the homing algorithm.

considered a single line, which reduces the allocation process by approximately 89%.

Figure 15(a) shows the two printheads cooperating to print the spiral layer. At the instant shown in the figure, the upper printhead is in the process of completing a green line, while the lower printhead has just finished its designated toolpath. As no further toolpath allocation is possible for the lower printhead at this point, the collision avoidance algorithm is activated. The algorithm then directs the lower printhead towards its home position. It's worth noting, however, that sending it directly to the home position would risk a collision. Therefore, a modified sequence is executed, directing the lower printhead to a point to the left of the home position first, thereby avoiding any collision.

The second case features a complex layer from a topology-optimised bracket [47], with layer dimensions of 184 mm × 60 mm. Figure 15(b) shows the result of

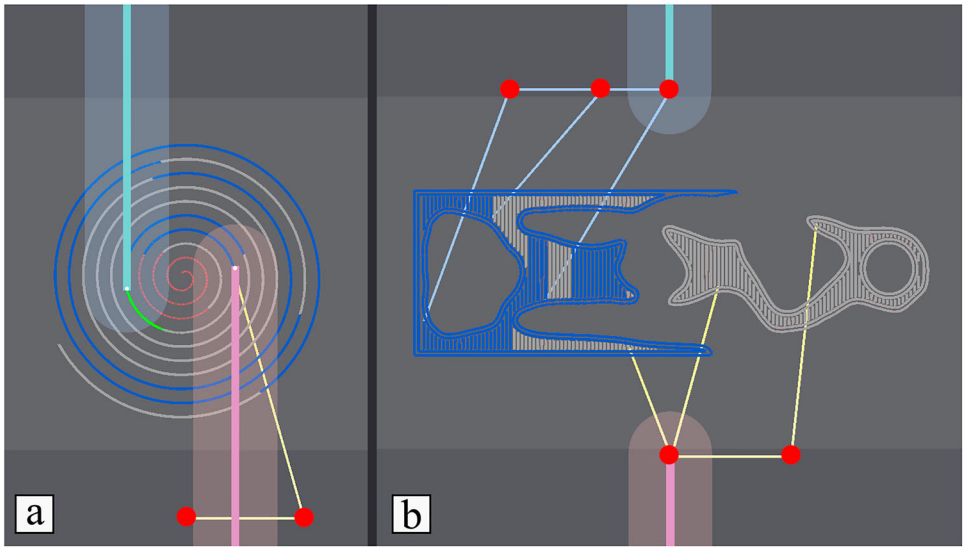

Figure 15. Two layers to evaluate the collision avoidance algorithm (a) The middle of a spiral layer in which the lower printhead is sent to the home position to avoid collisions, (b) fully printed bracket layer in which the collision avoidance algorithm was activated four times.

printing the layer by two mobile robots. Due to the high width-to-length ratio of 3.06, the percentage of positive collision checks was significantly reduced, averaging 62% less than the spiral layer. Despite the low frequency of potential collisions, the collision avoidance algorithm was activated four times in Figure 15(b), twice for each printhead. Furthermore, the algorithm infrequently employed the horizontal ray; in nearly 95% of the simulations, the printheads were either directed straight to their home positions or to nearby points, minimising the need for more complex collision avoidance moves.

5.2. Experimental works

5.2.1. Setup

Before starting the printing process, the XY-axes calibration and Z-homing are performed. The XY-axes calibration was implemented in two steps: printing platform location approximation and deviation correction in the XY axes. The mobile robots' locations are known using the developed localisation system. However, the location of the printing platform is unknown. Therefore, the printing platform location was found by using a third Vive tracker that was fixed at a known distance from the printing platform. The third Vive tracker collects position data for a period of 5 s and subsequently generates an estimated position by calculating the average of all the data points collected within this time frame. This results in a mean position estimate that represents the centre of the sampled locations. Through this method, it was possible to reduce the impact of individual measurement errors and to provide a more accurate estimation of the tracker position. Therefore, the position of the centre of the printing platform was calculated knowing the third tracker position and its distance to the printing platform. However, this calculation was not free of errors, thus the two mobile robots were asked to print a square on the generated centre, and the deviations on the X and Y axes were measured and then corrected in the software.

The Z-homing makes use of the current control mode of the Z-axis motors and the knowing distance between the mobile robot platform and the printing platform surface in the Z-axis. Both mobile robots are commanded to use the current control mode to send the Z-axis to its lowest possible position that touches the platform of the mobile robot. After that, the Z-axis motors use the position control mode and move the Z-axis to the position that matches the height of the surface of the printing platform.

5.2.2. Cooperative printing

The developed system was used to print two one-layer objects with different geometrical features. The objects were sliced in Cura using the following slicing settings: 1.4 mm line width, 2 walls, 0° and 90° rectilinear patterns, 225°C and 55°C hotend and hotbed temperatures, respectively. The mobile robots move at 18 mm/s coast speed and the safety distance around the printheads, and the arms is 45 mm. All objects were printed using polylactic acid (PLA) in a filament form (Prusa, Czech Republic). The lower mobile robot was fed a white PLA filament while the upper was fed a blue one, which allowed the visualisation of the effect of cooperation between the robots.

The first object (Figure 16) is a 150 mm × 95 mm rectangle with two rectangular holes of 65 mm × 33 mm centred on the left and right of the object. The slicer view of the object showing the inside print lines in yellow and the outer print lines in green and red are shown in Figure 16(a). To test the capabilities of the developed system, the mobile robots cooperatively printed the same part three times (Figure 16(b–d)). The variations between the toolpaths in the three cases show the randomness involved in the toolpath allocation as well as the effect of the unpredictable motion of the mobile robots. This object is printed in 11.5 min if a single-printhead machine is used. However, our cooperative mobile robots took 6.1, 7.25, and 6.93 min to print the parts, resulting in a 47%, 40%, and 37% reduction in the total printing time. The first printing test (Figure 16(b)) showed the highest amount of cooperation between the robots as can be clearly seen in the ratio of blue to white materials, resulting in the fastest printing time. The main reason of the high efficiency is the symmetrical toolpath allocation which resulted in even distribution of the workload between the two mobile robots. In the second test (Figure 16(c)), the lower mobile robot printed around 40% more than the upper mobile robot, which resulted in a slower printing process. Finally, in the third test (Figure 16(d)) both mobile robots printed almost the same amount of print lines. However, the travelling distance of both mobile robots is very large, as can be seen in Figure 17. The diagonal print lines in the middle of Figure 16(c) and the top right corner of Figure 16(d) are due to filament oozing during the travel movement, as can be seen when inspecting the toolpaths in Figure 17.

The second object (Figure 18) is a wide hexagonal part with a slot in the middle, and total dimensions of 155 mm × 100 mm. Figure 18(a) shows the slicer view and the print lines and Figure 18(b) shows the actual printed part by the mobile robots. The average printing time reduction by using the developed system is around 46%. A higher reduction in printing time can be achieved by combining both offline and online path planning. In the case of this object, the initial toolpath

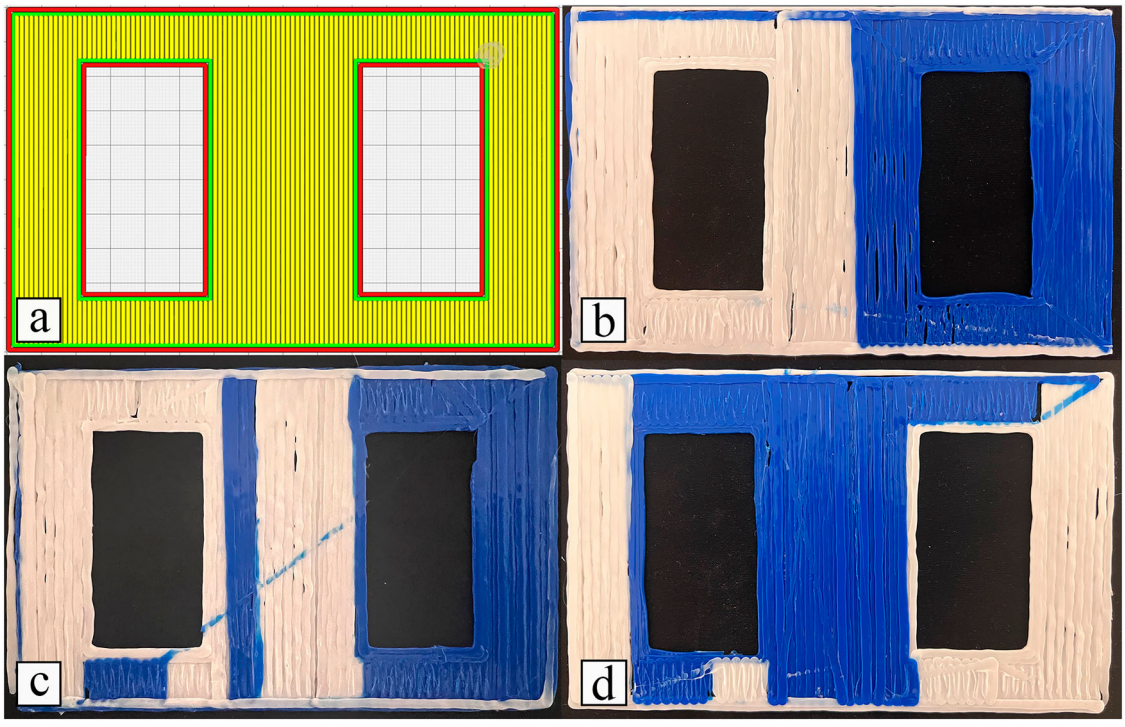

Figure 16. A one-layer object with two holes (a) slicer view, (b–d) multiple attempts of cooperatively printing the layer using the two mobile robots.

allocation was designated using an offline algorithm which was repeated until a low printing time was found. Then, the mobile robots perform the offline found toolpaths and request from the online path planner real-time corrections for problematic toolpaths.

Offline path planning alone is not reliable for the mobile robots as toolpaths generated are not always collision-free when performed by the mobile robots. Figure 19(a) shows the offline path planning of the hexagonal part, while Figure 19(b) represents the actual motion of the mobile robots in real time. The toolpaths executed are not in collision, but as can be seen from the top of the figure, the location of the lower printhead predicted by the offline path planner is incorrect. The offline path planner predicted that the lower mobile robot would be above the centre of the slot, while the actual motion of the lower mobile is near the bottom of the part, which corresponds to around 60 mm of error. This deviation did not produce any collision as the two printheads are far from each other. However, the next toolpaths allocated by the offline path planner result in collisions (bottom of Figure 19(a)) as it requests the upper printhead to print the inner line of the top left corner of the hexagon. Thus, the online path planner sent the upper mobile robot to its home position using the homing algorithm to avoid the collision as can be seen at the bottom of Figure 19(b). Without real-time monitoring and planning, the two mobile robots would collide.

Additionally, the online path planning and the mobile robots were also used to cooperatively print a multi-layer object. The object printed is a rectangular prism with dimensions of 160 mm × 100 mm × 10 mm. The 3D model was sliced with Cura considering 1 mm as layer height settings, 15% infill density, 2 bottom solid layers, and 3 top solid layers. Figure 20 shows the cooperatively printed rectangular prism. The total printing time required to print this part using the cooperative mobile robots system was 56.1 min. Therefore, as a single-printhead mobile robot requires 1.6 h to print it, the developed cooperative system increases the printing speed by around 71.1% without changing the translation speed of the robots. During the cooperative printing, the lower and upper mobile robots were homed 11 and 12 times, respectively. This indicates that a total of 23 potential collisions would have occurred without the homing algorithm. In such cases, the mobile robots would have collided, rendering them incapable of completing the printing task. Figure 21 shows the printing process of the rectangular prism. On the right, the lower mobile robot is waiting at its home position.

5.3. Limitations

As this work is the first to explore online path planning for cooperative printing systems, it is also subjected to some limitations that must be addressed in the future.

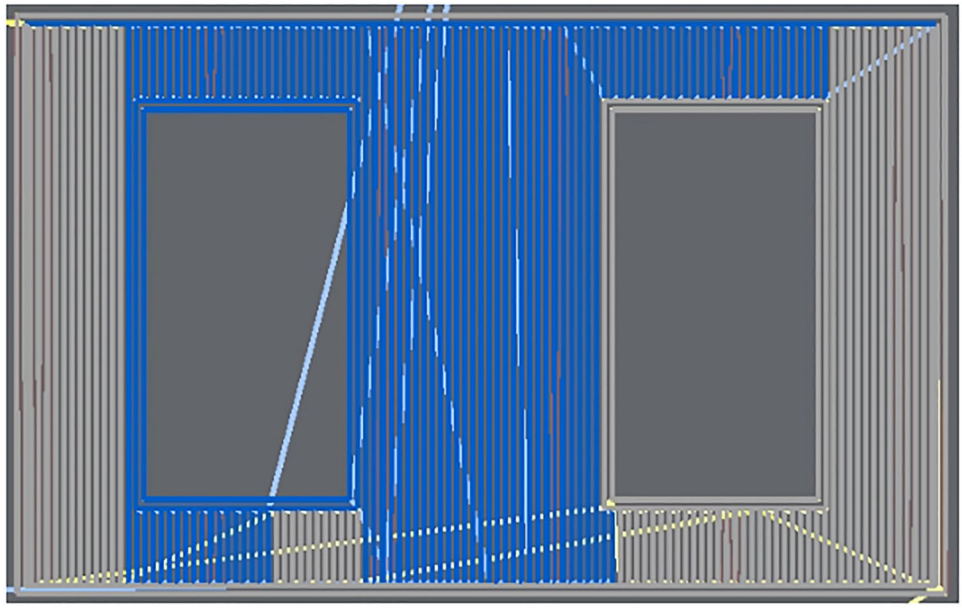

Figure 17. Visualisation of the toolpaths of the third attempt in the view of the software (Figure 16(d)).

These limitations are mainly related to performance specification, the collision avoidance algorithm, and the kinematic configuration of the system.

As discussed in Section 5.1.2, online path planning has an inherent limitation in the sense that no minimum performance can be specified. This is because the relevant data are not pre-known but must be collected in real-time during the printing process. Another limitation arises from our toolpath allocation algorithm, which prevents even the starting allocation performance from being specified. However, this issue can potentially be addressed by modifying the algorithm, as the initial allocation information can be known before the printing starts. Due to this limitation, several results presented earlier exhibited reduced printing time efficiencies due to asymmetric allocations. An amendment to the algorithm could embed symmetrical allocation, thereby altering the toolpath allocator's behaviour to distribute tasks more evenly between the printheads.

While our collision avoidance algorithm has demonstrated success in averting collisions across numerous physical and virtual test scenarios, it is not without drawbacks. Any time the printheads spend in the home position constitutes as lost time, reducing the overall efficiency of the cooperative printing task. As an example, in the multi-layer print discussed in Section 5.2.2, the printheads were sent to their home positions 23 times, cumulatively costing about 5 min. Eliminating this would boost efficiency from 71.1% to 88.2%. Thus, optimising the homing algorithm is needed for future applications.

The main assumption behind the developed path planning suite is that the system kinematic configuration is designed as multi-arm [1]. Therefore, only systems able to be configured as multi-arm can implement the

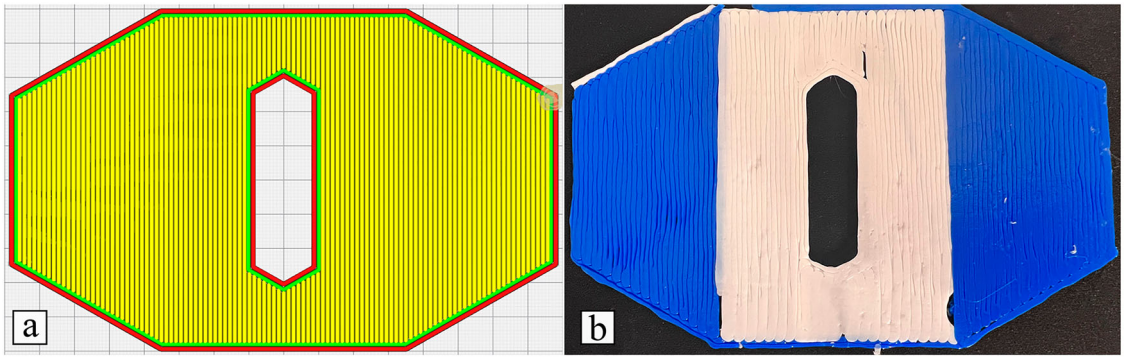

Figure 18. Wide hexagonal part with a hole in the middle (a) slicer view; (b) cooperatively printed part using the two mobile robots.

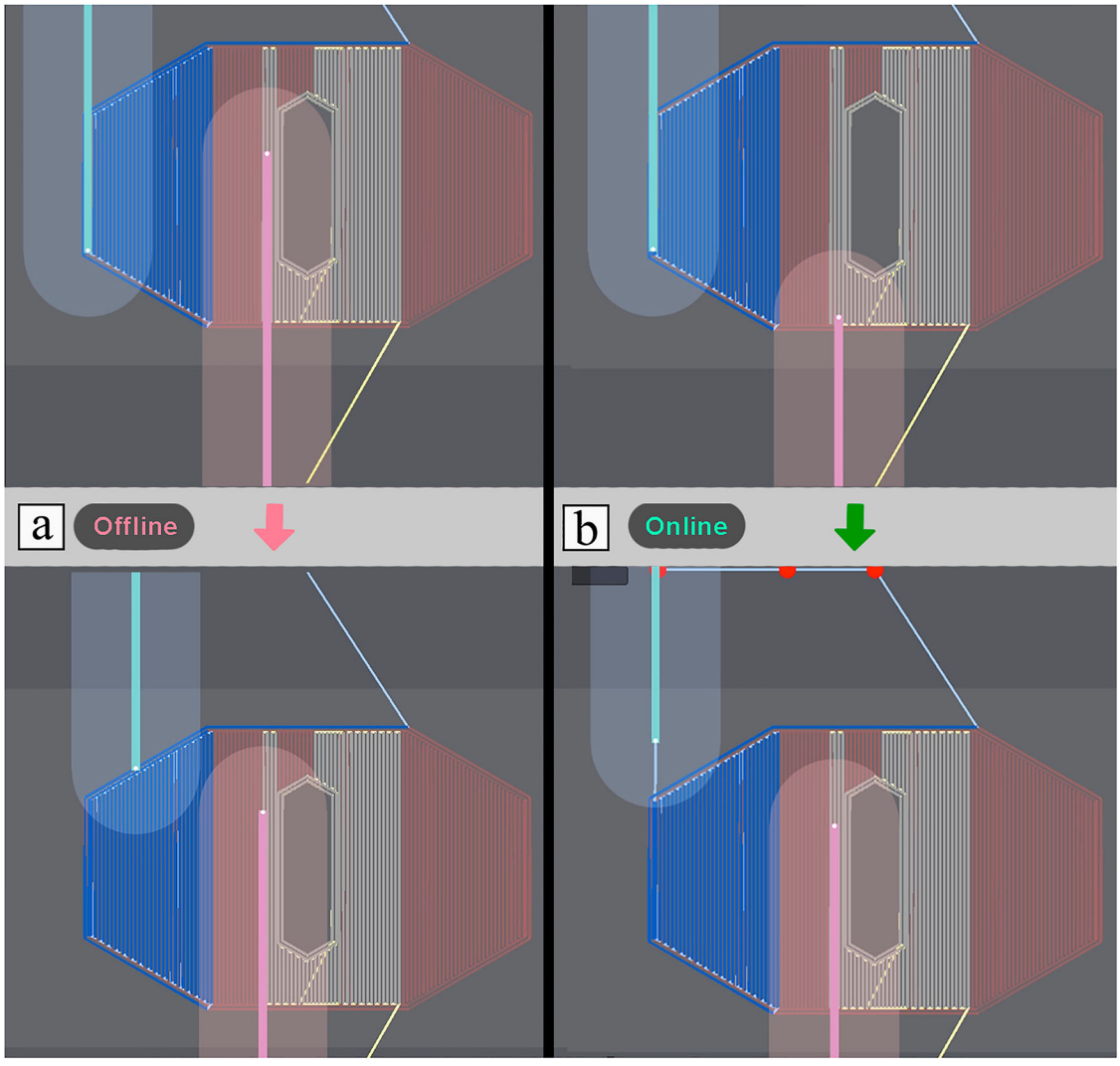

Figure 19. A problematic toolpath that results in a collision if the actual mobile robots followed the offline path planner. (a) Offline path planning; (b) online path planning and the actual motion of the robots. Top is the previous timestep and bottom is the next timestep. The offline path planning (a) predicted not collision would occur, however the online path planner updated the allocation as the offline prediction was actually wrong.

algorithms developed in this research (see Figure 22 for two possible systems). While our toolpath allocation algorithm can be readily adapted to other configurations, both the collision avoidance and collision checking algorithms are specifically designed for multi-arm setups, where the printhead-carrying link is assumed to be rigid and fixed to a base behind the printing platform, i.e. not a multi-joint robotic arm.

6. Conclusions and future works

Cooperative printing is an emerging concept in additive manufacturing that significantly increases fabrication speed. However, this approach still presents several challenges. Due to the uncertainty in predicting the exact state of a printhead, offline path planning is unreliable. In the literature, this issue has been addressed by using systems with very high acceleration and robotic systems that have easy-to-predict behaviours such as gantry systems. In the case of low acceleration and high-speed systems as well as error-prone systems such as mobile robots, offline path planning is impractical since it assumes that the system moves in a predictable manner that is not affected by external and internal disturbances. Thus, we developed path planning algorithms for cooperative printing systems that are designed for online applications. For this purpose, a motion model was developed for the mobile robots in the system, which is used by the toolpath allocation algorithm and considers the real-time state of the robots when allocating toolpaths. Furthermore, as the path planning is performed online, we developed both collision checking

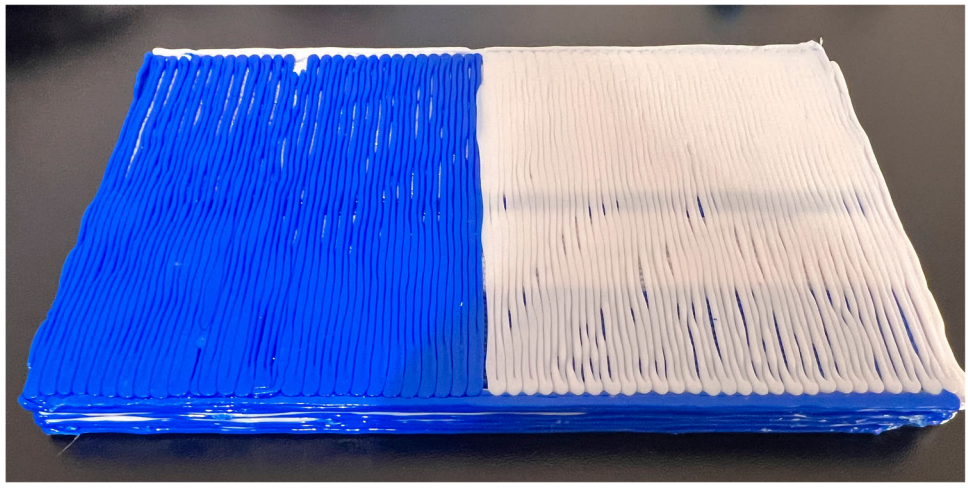

Figure 20. The bottom view of the multi-layer rectangular prism printed by the mobile robots cooperatively.

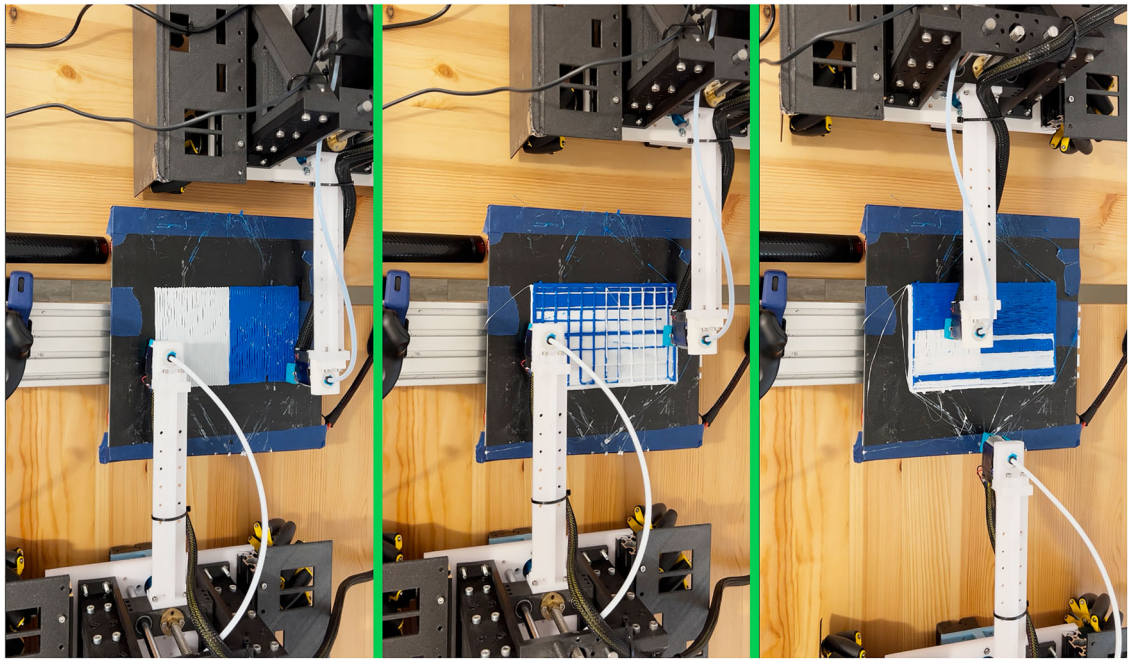

Figure 21. Sequence of images showing the printing process of the rectangular prism. Left is bottom layer printing; middle is infill printing; and right is the top layer.

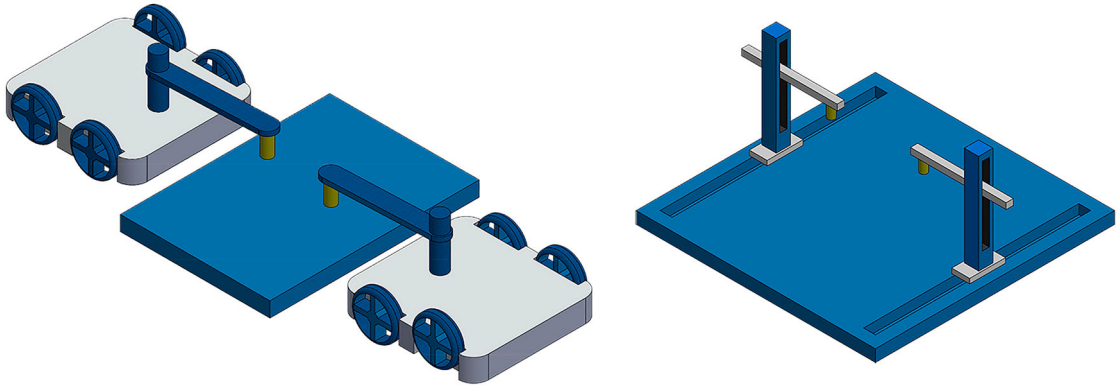

Figure 22. Two different systems implementing the multi-arm configuration.

and avoidance algorithms that were implemented in real time. Contrary to offline path planning, a toolpath allocated to one robot cannot be replanned if it resulted in future toolpaths that produce collisions. Thus, we developed a collision avoidance algorithm that guarantees collision avoidance in such cases.

The developed system was tested both virtually and physically. For the experimental work, two accurate mobile robots, capable of performing filament-based extrusion additive manufacturing, were designed, and implemented. The developed online path planning algorithms were compared to a similar system but performed fully offline algorithms. It was found that online algorithms cannot specify a specific performance, but the computation speed of the online algorithms was faster. Furthermore, it was shown that offline path planning results in collisions not accounted for when the actual mobile robots start the cooperative printing task. However, those collisions were fully avoided when using the developed online system. Finally, the cooperative printing system was physically verified by printing several single layers and a multi-layer object. The system was shown to be capable of reducing the printing time for AM by up to 47%.

This work serves as a foundational step in advancing online path planning for cooperative printing systems, opening the door for future research to refine and extend these initial efforts. Though our toolpath allocation and collision avoidance algorithms demonstrate promise, they are not without limitations. Future work will focus on enhancing the efficiency of toolpath allocation and minimising the time spent at the home position. In addition, while the current path planner is designed for a two-printhead system, subsequent research will explore its adaptability to environments with more than two cooperating robots. Additionally, during the printhead's return to the home position, potential oozing of filament may occur due to maintained heat. Consequently, future research will investigate automated approaches to pre-emptively mitigate oozing, such as prime line print post-homing or utilising retraction features to manage excess filament effectively.

Based on this research, two significant applications will be explored in the future. First, the work is aimed to impact a broader range of additive manufacturing domains, including construction and concrete 3D printing, where the large-scale and time-sensitive nature of such large projects can significantly benefit from the efficiencies of cooperative printing. Additionally, swarm manufacturing, where numerous robots operate in a single environment, is a largely unexplored area for online path planning. Existing systems in swarm manufacturing rely on offline planning; however, the complexity and scale of these operations make them ideal candidates for real-time planning and correction.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

The authors wish to acknowledge the support from UKRI Interdisciplinary Circular Economy for Textiles: Circular Bioeconomy for textile materials [Project code: EP/V011766/1].

Data availability statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Abdullah Alhijaily http://orcid.org/0000-0001-6518-9826

References

[1] Alhijaily A, Kilic ZM, Bartolo ANP. Teams of robots in additive manufacturing: a review. Virtual Phys Prototyp. 2023;18(1):2162929. doi:10.1080/17452759.2022.2162929

[2] Ansari AA, Kamil M. Effect of print speed and extrusion temperature on properties of 3D printed PLA using fused deposition modeling process. Mater Today Proc. 2021;45:5462–5468. doi:10.1016/j.matpr.2021.02.137

[3] Rezaeian P, Ayatollahi MR, Nabavi-Kivi A, et al. Effect of printing speed on tensile and fracture behavior of ABS specimens produced by fused deposition modeling. Eng Fract Mech. 2022;266:108393. doi:10.1016/j.engfracmech.2022.108393

[4] Siciliano B, Sciavicco L, Villani L, et al. *Robotics: modelling, planning and control*. 1st ed. London: Springer; 2009.

[5] Bhatt PM, Kabir AM, Peralta M, et al. A robotic cell for performing sheet lamination-based additive manufacturing. Addit Manuf. 2019;27:278–289. doi:10.1016/j.addma.2019.02.002

[6] Wu C, Dai C, Fang G, et al. RoboFDM: a robotic system for support-free fabrication using FDM. 2017 IEEE International Conference on Robotics and Automation (ICRA) 2017. p. 1175–1180. doi:10.1109/icra.2017.7989140

[7] Dörfler K, Dielemans G, Lachmayer L, et al. Additive manufacturing using mobile robots: opportunities and challenges for building construction. Cem Concr Res. 2022;158:106772. doi:10.1016/j.cemconres.2022.106772

[8] Sustarevas J, Kanoulas D, Julier S. Autonomous mobile 3D printing of large-scale trajectories. 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2022. p. 6561–6568. doi:10.1109/IROS47612.2022.9982274

[9] Zhang X, Li M, Lim JH, et al. Large-scale 3D printing by a team of mobile robots. Autom Constr. 2018;95:98–106. doi:10.1016/j.autcon.2018.08.004

[10] Troemner M, Ramyar E, Meehan J, et al. A 3D-printing centered approach to Mars habitat architecture and fabrication. J Aerosp Eng. 2022;35(1):04021109. doi:10.1061/(ASCE)AS.1943-5525.0001359

[11] Khamis A, Hussein A, Elmogy A. Multi-robot task allocation: a review of the state-of-the-art. In: Koubâa A, Martínez-de Dios J, editors. *Cooperative robots and sensor networks 2015. Studies in computational intelligence*. Cham: Springer; 2015. p. 31–51. doi:10.1007/978-3-319-18299-5_2

[12] Poudel L, Marques LG, Williams RA, et al. Toward swarm manufacturing: architecting a cooperative 3D printing system. J Manuf Sci Eng. 2022;144(8):81004. doi:10.1115/1.4053681

[13] Krishnamurthy V, Poudel L, Ebert M, et al. Layerlock: layer-wise collision-free multi-robot additive manufacturing using topologically interlocked space-filling shapes. Comput Aided Des. 2022;152:103392. doi:10.1016/j.cad.2022.103392

[14] Bhatt PM, Nycz A, Gupta SK. Optimizing multi-robot placements for wire arc additive manufacturing. 2022 International Conference on Robotics and Automation (ICRA). 2022. p. 7942–7948. doi:10.1109/ICRA46639.2022.9812318

[15] Xu X, Wang Z, Feng C. Projector-guided non-holonomic mobile 3D printing. 2021 IEEE International Conference on Robotics and Automation (ICRA). 2021. p. 8039–8045. doi:10.1109/ICRA48506.2021.9561719

[16] Zhang K, Chermprayong P, Xiao F, et al. Aerial additive manufacturing with multiple autonomous robots. Nature. 2022;609(7928):709–717. doi:10.1038/s41586-022-04988-4

[17] Cai Y, Choi SH. Deposition group-based toolpath planning for additive manufacturing with multiple robotic actuators. Proc Manuf. 2019;34:584–593. doi:10.1016/j.promfg.2019.06.223

[18] Jiang Z, Wang H, Sun Y. Improved co-scheduling of multi-layer printing path scanning for collaborative additive manufacturing. IISE Trans. 2020;53:960–973. doi:10.1080/24725854.2020.1807076

[19] Bui H, Pierson HA, Pinkley SN, et al. Toolpath planning for multi-gantry additive manufacturing. IISE Trans. 2020;53(5):552–567. doi:10.1080/24725854.2020.1775915

[20] Jin Y, Pierson HA, Liao H. Toolpath allocation and scheduling for concurrent fused filament fabrication with multiple extruders. IISE Trans. 2019;51(2):192–208. doi:10.1080/24725854.2017.1374582

[21] Wang Y, Gu Z, Song L, et al. Speeding up 3D printing using multi-head slicing algorithms. 2017 5th International Conference on Enterprise Systems (ES). 2017. p. 99–106. doi:10.1109/es.2017.23

[22] Zhang J, Khoshnevis B. Optimal machine operation planning for construction by contour crafting. Autom Constr. 2013;29:50–67. doi:10.1016/j.autcon.2012.08.006

[23] Titan Robotics, Cronus: Cronus | Titan robotics, 2017 [cited 2023 May 10]. Available from: https://titan3drobotics.com/tag/cronus/

[24] Project Escher: Autodesk. 2017 [cited 2023 May 10]. Available from: https://theindexproject.org/award/nominees/1931

[25] Tianying H, Shengfu Y, Anguo H, et al. Path planning and forming of wire multi-arc additive collaborative manufacture for marine propeller bracket. J Manuf Process. 2021;68:1191–1201. doi:10.1016/j.jmapro.2021.06.028

[26] Urhal P, Weightman A, Diver C, et al. Robot assisted additive manufacturing: a review. Robot Comput Integr Manuf. 2019;59:335–345. doi:10.1016/j.rcim.2019.05.005

[27] Bhatt PM, Kabir AM, Malhan RK, et al. A robotic cell for multi-resolution additive manufacturing. 2019 International Conference on Robotics and Automation (ICRA). 2019. p. 2800–2807. doi:10.1109/icra.2019.8793730.

[28] Shen H, Pan L, Qian J. Research on large-scale additive manufacturing based on multi-robot collaboration technology. Addit Manuf. 2019;30. doi:10.1016/j.addma.2019.100906

[29] Sustarevas J, Benjamin Tan KX, Gerber D, et al. YouWasps: towards autonomous multi-robot mobile deposition for construction. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2019. p. 2320–2327. doi:10.1109/IROS40897.2019.8967766.

[30] Poudel L, Elagandula S, Zhou W, et al. Decentralized and centralized planning for multi-robot additive manufacturing. J Mech Des. 2023;145(1):12003. doi:10.1115/1.4055735

[31] Zhao Y, Zheng Z, Liu Y. Survey on computational-intelligence-based UAV path planning. Knowl Based Syst. 2018;158:54–64. doi:10.1016/j.knosys.2018.05.033

[32] Huang J, Chen Q, Jiang H, et al. A survey of design methods for material extrusion polymer 3D printing. Virtual Phys Prototyp. 2020;15(2):148–162. doi:10.1080/17452759.2019.1708027

[33] Nayyeri P, Zareinia K, Bougherara H. Planar and nonplanar slicing algorithms for fused deposition modeling technology: a critical review. Int J Adv Manuf Technol. 2022;119(5–6):2785–2810. doi:10.1007/s00170-021-08347-x

[34] Bui H, Pierson HA, Nurre SG, et al. Tool path planning optimization for multi-tool additive manufacturing. Proc Manuf. 2019;39:457–464. doi:10.1016/j.promfg.2020.01.389

[35] Wang Y, Gu Z, Song L, et al., eds. Speeding up 3D printing using multi-head slicing algorithms. International Conference on Enterprise Systems (ES); 22–24 Sept. 2017. doi:10.1109/ES.2017.23

[36] Elagandula S, Poudel L, Sha Z, et al. Multi-robot path planning for cooperative 3D printing. Volume 1: Additive Manufacturing; Advanced Materials Manufacturing; Biomanufacturing; Life Cycle Engineering; Manufacturing Equipment and Automation. 2020. doi:10.1115/MSEC2020-8390

[37] Luis CE, Vukosavljev M, Schoellig AP. Online trajectory generation with distributed model predictive control for multi-robot motion planning. IEEE Robot Autom Lett. 2020;5(2):604–611. doi:10.1109/LRA.2020.2964159

[38] McPherson J, Zhou W. A chunk-based slicer for cooperative 3D printing. Rapid Prototyp J. 2018;24(9):1436–1446. doi:10.1108/RPJ-07-2017-0150

[39] Keating SJ, Leland JC, Cai L, et al. Toward site-specific and self-sufficient robotic fabrication on architectural scales. Sci Robot. 2017;2(5):8986. doi:10.1126/scirobotics.aam8986

[40] Leng C, Cao Q, Huang Y. A motion planning method for omnidirectional mobile robot based on the anisotropic characteristics. Int J Adv Rob Syst. 2008;5(4):45. doi:10.5772/6228

[41] Czyzewski P, Marciniak D, Nowinka B, et al. Influence of extruder's nozzle diameter on the improvement of functional properties of 3D-printed PLA products. Polymers. 2022;14(2). doi:10.3390/polym14020356

[42] Borenstein J, Everett HR, Feng L, et al. Mobile robot positioning: sensors and techniques. J Robot Syst. 1997;14(4):231–249. doi:10.1002/(SICI)1097-4563(199704)14:4<231::AID-ROB2>3.0.CO;2-R

[43] Thrun S, Burgard W, Fox D. *Probabilistic robotics*. 1st ed. Cambridge, MA: The MIT Press; 2005.

[44] Alhijaily A, Kilic ZM, Bartolo P. Localization and control of a mobile robot for additive manufacturing. Progress in digital and physical manufacturing. Springer tracts in additive manufacturing. 2023. p. 72–81. doi:10.1007/978-3-031-33890-8_7

[45] Lynch KM, Park FC. *Modern robotics*. 1st ed. Cambridge: Cambridge University Press; 2017.

[46] Glover F, Laguna M. Tabu search. In: Du D, Pardalos P, editors. *Handbook of combinatorial optimization*. Boston: Springer; 1998. p. 2093–2229. doi:10.1007/978-1-4613-0303-9_33.

[47] Chagnot M. Bracket topology optimization. 2019 [cited 2023 Sep 2]. Available from: https://grabcad.com/library/bracket-topology-optimization-2-1