Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 00.0000/ACCESS.0000.0000000

Swarm-Based Trajectory Generation and Optimization for Stress-Aligned 3D Printing

XAVIER GUIDETTI[1,2], EFE C. BALTA[1,2], (Member, IEEE), and JOHN LYGEROS[1], (Fellow, IEEE)
[1]Automatic Control Laboratory, ETH Zurich, Physikstrasse 3, 8092, Zurich, Switzerland (e-mail: {xaguidetti,jlygeros}@control.ee.ethz.ch)
[2]Inspire AG, Technoparkstrasse 1, 8005, Zurich, Switzerland (e-mail: efe.balta@inspire.ch)

Corresponding author: Efe C. Balta (e-mail: efe.balta@inspire.ch).

Research supported by Innosuisse (project №102.617 IP-ENG) and by the Swiss National Science Foundation under NCCR Automation (grant №180545)

ABSTRACT In this study, we present a novel swarm-based approach for generating optimized stress-aligned trajectories for 3D printing applications. The method utilizes swarming dynamics to simulate the motion of virtual agents along the stress produced in a loaded part. Agent trajectories are then used as print trajectories. With this approach, the complex global trajectory generation problem is subdivided into a set of sequential and computationally efficient quadratic programs. Through comprehensive evaluations in both simulation and experiments, we compare our method with state-of-the-art approaches. Our results highlight a remarkable improvement in computational efficiency, achieving a $115\times$ faster computation speed than existing methods. This efficiency, combined with the possibility to tune the trajectories spacing to match the deposition process constraints, makes the potential integration of our approach into existing 3D printing processes seamless. Additionally, the open-hole tensile specimen produced on a conventional fused filament fabrication set-up with our algorithm achieve a notable $\sim 10\%$ improvement in specific modulus compared to existing trajectory optimization methods.

INDEX TERMS 3D printing, additive manufacturing, fused filament fabrication, stress-aligned printing, swarming, trajectory optimization

I. INTRODUCTION

In 3D printing, a desired three-dimensional part is created by depositing material in a layer-wise fashion. This family of manufacturing processes, also known as Additive Manufacturing (AM), enables the creation of complex geometries and has been steadily gaining popularity in the last decades. The most widely AM technique is Fused Filament Fabrication (FFF) [1], which is also known as Fused Deposition Modeling (FDM) or Material Extrusion Additive Manufacturing [2]. In FFF, a numerically controlled heated extruder moves along a predefined trajectory while depositing melted plastic, forming the desired part. Typically, the geometry to be manufactured is first sliced into equally spaced planes (known as *planar* printing) [3] or two-dimensional manifolds (known as *non-planar* printing) [4], [5]. Then, trajectories are created inside each slice. This process is generally achieved via a *slicer* software, which mostly focuses on generating trajectories that form the part in a rapid or precise manner. Depending on the used material, desired geometry, printing equipment, and required geometrical accuracy, the slicing and trajectory generation process can be adapted to optimize the properties of the manufactured part [6]–[9].

The recent developments of high performance materials, paired with a general technological improvement of the printing process, has enabled the usage of parts produced with FFF in a wide range of applications where high strength and stiffness are required [10], [11]. Motivated by these developments, recent works have focused on generating stress-aligned printing trajectories that maximize the mechanical properties of FFF printed parts, both in the planar [12] and non-planar case [13], [14]. Other works have tackled the same problem for the specific case of fiber-reinforced filaments, which must be printed in a continuous fashion [15]–[17]. The general idea in this field of research is that, for a given material, manufacturing process, and geometry, there exists a stress-aligned trajectory that produces the desired part while improving its strength and stiffness under a general or specific load case. Finding such a trajectory is a challenging task due to complex part features, manufacturing process constraints and geometrical accuracy. The surveyed works that produce acceptable results in practice generally compute print trajectories by solving a very large optimization problem, which is complex and time-consuming. Additionally, these methods produce a single solution which cannot be tuned or refined to exactly match the printing process properties and mechanical requirements.

In this work, we propose a novel approach to the generation of stress-aligned print trajectories that is based on swarming dynamics. Each computed print line is the trace of a simulated agent moving through the part to be printed. Agents move in a swarm, and their motion is dictated by the stress generated by the part's load case, producing stress-aligned trajectories. By utilizing a swarm-based approach, the complex stress-aligned trajectory generation problem is broken down into a set of sequential and computationally efficient optimization problems. This significantly reduces the total time required for optimization. Additionally, the swarming dynamics enable great flexibility: our swarm-based trajectory generation method can be fine-tuned to produce trajectories that perfectly match the printing process limitations. The main contributions of this work are:

- A novel, computationally efficient, and flexible swarm-based approach to stress-aligned trajectory generation for 3D printing,
- A study of the methods' behavior with different settings, and
- The experimental comparison of swarm-based stress-aligned trajectory generation with a state-of-the-art method.

In Section II we introduce the methods of which we make use in our approach and discuss related literature. Section III details the swarm-based trajectory generation method. In Section IV we analyze the performance of our method in detail and benchmark it. Section V concludes the paper.

II. BACKGROUND

A. FINITE ELEMENT ANALYSIS

Finite Element Analysis (FEA) is a very well known and widely used method to simulate numerically the stress field inside a loaded geometry [18]. We utilize it to compute the stress along which the printing trajectories for the optimized part will be aligned. In FEA, the part to be manufactured is first subdivided in a tetrahedral mesh. Then the forces produced by the predefined load case are added to the problem as boundary conditions. After assigning to the part its mechanical properties (i.e. Young's modulus and Poisson ratio), an FEA solver is used to find a numerical solution to the resulting set of differential equations. Solving this discretized problem corresponds to finding a Cauchy stress tensor $\boldsymbol{\sigma}$ for each node in the mesh. Typically, the Cauchy stress tensor is decomposed using eigenvalue decomposition to obtain the principal stresses and principal directions [19]. Now, the stress tensor can be represented as a diagonal matrix in a reference frame oriented along the principal directions of the decomposition. The diagonal entries σ_1, σ_2 and σ_3 of the principal stress matrix are ordered such that $|\sigma_1| \geq |\sigma_2| \geq |\sigma_3|$. In this work, we only consider the principal stress σ_1 and the corresponding normalized eigenvector $\mathbf{e}_1$. This eigenvector indicates the direction along which the principal stress acts. At any given node of the mesh, we indicate the product of the principal stress with the corresponding normalized eigenvector as $\mathbf{s} = \sigma_1 \mathbf{e}_1$, and we loosely call it *local principal stress vector*.

B. STRESS-ALIGNED PRINTING

The goal of stress-aligned printing is to optimize the internal structure of a 3D printed part to enhance its mechanical performance. The approach exploits the anisotropic nature of AM processes. In particular, parts produced with FFF have been shown to be anisotropic, as their mechanical properties are better in the direction in which the plastic filament has been deposited. This effect can be moderately strong with materials such as PolyLactic Acid (PLA) (see Fig. 2 from [13]) or extremely strong with materials such as Liquid Crystal Polymers (LCP) [20]. In practice, after conducting a FEA for a given geometry and load case, the field of local principal stress vectors is used to guide the trajectory generation process. The objective is to produce trajectories for which the local printing direction is aligned with the local principal stress. This has been shown to maximize the strength and stiffness of the final object under the given load case [3], [13], [21]. While the stress-alignment of trajectories has been the central goal of past works, it has been achieved at the expense of computational efficiency and of flexibility of the approach. Existing methods require long and complex computations, often relying on commercial optimization solvers, which makes them unsuitable for widespread use in conventional slicers. Furthermore, for a given geometry and load case, a unique trajectory is generated, with no possibility for refinement. This is particularly limiting, since the manufacturability of trajectories is a central issue in FFF. Generally, a stress-aligned trajectory for a complex part is characterized by a variable spacing between the print lines. However, the possibility to deposit lines of variable width is constrained by the material properties and the extrusion process. Attempting to print beyond these constraints necessarily produces over- or under-extrusion, strongly reducing the quality of the printed part and its mechanical properties [22], [23]. Thus, there exists a need for a computationally efficient stress-aligned trajectory generation algorithm that can be tuned to produce trajectories with a desired distribution of line spacing, ensuring manufacturability. This last aspect is becoming increasingly relevant as recent works have enhanced the quality and width range of variable width FFF [24], [25].

C. SWARMING

Numerous methods for swarm formation, modeling, and navigation for multiple autonomous agents have been proposed in the literature [26]–[28]. Often, the interactions between individuals (or between individuals and the space surrounding them) have been modeled using artificial potential functions. The main applications driving this line of research have been robot navigation and control [29]–[31] or multi-agent coordination [32]. Interestingly, however, the inspiration for a more general approach to the study of swarm aggregation came from mathematical biology [33], [34]. Gazi and Passino [35]–[37] have introduced a class of attraction and repulsion functions between individuals that ensure the aggregation of a swarm moving through an environment, prevent individuals from making contact with each other, and allow for formation control. They have further extended their work to follow an energy approach [38]. This method, mimicking the behavior of swarms in nature, proposes that the motion of individual agents is dictated by a *biological potential energy* to be minimized. The total energy of a swarm is given by the sum of its kinetic and potential energies. Trivially, a swarm kinetic energy corresponds to the sum of the kinetic energies of its individual components, as defined in classical mechanics. The notion of potential energy of a swarm, however, has been extended beyond classical physical potential energy to also include aggregation, environment and predator potentials. Given these notions, it is possible to model a swarm by applying a Lagrangian approach to the swarm total energy. Intuitively, the agents composing a swarm have a tendency to maintain their kinetic energy unchanged and to attain minimal potential. A flock of birds or a school of fish, for example, seek to travel with constant speed and direction (thus avoiding kinetic energy variations), to maintain a comfortable and safe distance between individuals (minimizing the aggregation potential), to explore areas rich in food (minimizing the environment potential), and to scatter and flee in the presence of threats (minimizing the predator potential).

III. METHOD

The approach we propose originates from the fascinatingly simple idea that the motion trajectories of a swarm of virtual point-mass agents can be used as manufacturing trajectories. If the swarm is carried through a mechanically loaded part by the load-induced stress flow, the resulting manufacturing trajectories are then stress-aligned. Similarly to other existing approaches in the literature [13], [21], we separate the tasks of slicing the part and of generating trajectories on a slice. We assume that the slices are given and only consider the trajectory generation problem. Irrespective of whether slices are planar or not, the trajectories are thus generated over a 2D manifold.

The generation of trajectories for FFF has specific requirements and peculiarities that make the existing approaches to swarm modeling not immediately usable for the task. The method introduced in Section II-C must be modified or extended in three areas:

1) agents in the swarm must avoid moving through existing trajectories,
2) the set of trajectories must cover the entire part, and
3) agents can be added to or removed from the swarm at will.

Condition 1 is not enforced in classical swarm modeling, since physical agents must simply avoid collisions among each other. Only the present location of agents is relevant for planning, and agents are allowed to cross existing trajectories. In swarming for manufacturing, however, both a spatial and temporal separation must be maintained between agents: the manufacturing trajectories cannot overlap as depositing material twice at the same location creates defects in the part.

Condition 2 ensures that the outside of the part produced with the generated trajectories matches the desired geometry, and that no voids are left inside the part. Additionally, in FFF (and in AM in general) the rate of material deposition is bounded. For this reason, good quality coverage of the part is achieved when the spacing between neighboring manufacturing trajectories respects upper and lower bounds. If trajectories are too far apart the space in between cannot be entirely filled, if trajectories are too close together too much material will be deposited at the same location.

Finally, Condition 3 exploits the non-physical nature of agents in the manufacturing problem to simplify the part coverage problem (Condition 2). Contrarily to a swarm of physical agents, we can change the number of agents in the swarm at any time. For example, it is possible to *spawn* new agents (which will produce a new manufacturing trajectory) where other trajectories have diverged, or to *kill* existing agents (which will interrupt an existing manufacturing trajectory) where trajectories compress excessively.

A. SWARM-BASED TRAJECTORY PLANNING FOR MANUFACTURING

Our approach has been principally inspired by the notions of aggregation and environment potentials from [38], which we have adapted to the context of manufacturing. Let us consider a mechanically loaded part for which a FEA has been conducted as discussed in Section II-A. We create a swarm of agents at the location where the largest external force is applied to the part. At initialization, the agents are homogeneously spaced, and neighboring agents are at a predefined distance matching the nominal desired distance between FFF print lines. We define the swarm potential as

$$P = P_a + KP_e \, , \tag{1}$$

where P_a and P_e are the aggregation and environment potentials of the swarm, and K is a tuning constant. P_a encodes the quality of the swarm formation: it is minimal when the spacing between neighboring agents corresponds to a predefined desired distance, and increases in case of spacing deviations. P_e quantifies the stress alignment of the agents' motion: it is minimal when agents follow the stress flow exactly, and increases when they deviate from it. To have the swarm draw a well-spaced set of stress-aligned trajectories through the part, we propose to utilize Algorithm 1.

B. METHODOLOGICAL DETAILS

Algorithm 1 is obviously only an outline of the method, that we first introduce in a simplified manner for clarity. In this section, we describe every required detail that composes our method for swarm-based trajectory planning for manufacturing.

Algorithm 1 Swarm-Based Trajectory Generation

```
Require: Agents initial location, Step size, Agents desired
    distance, FEA simulation, $K$
 1: while Part is not fully covered do
 2:     Advance each agent by one step size along the local
        principal stress direction
 3:     Reposition agents by minimizing $P$
 4:     if Two agents are too close together then
 5:         Kill one of them
 6:     end if
 7:     if Two agents are too far apart then
 8:         Spawn one agent in between
 9:     end if
10: end while
```

1) Step Direction and Size

The solution to an FEA simulation of stresses in a mechanical component produces a stress flow in the part. As the principal stress vectors forming the stress flow originate from the eigenvectors of the Cauchy stress tensor, only their direction is relevant to the stress alignment problem, while their orientation is not. Intuitively, local alignment between a print line and a stress vector is left unchanged by flipping the vector orientation. To produce a stress flow free from this $180°$ *ambiguity* (i.e. heterogeneous stress vectors orientations), past approaches have used simulated annealing [39] and rectification along the main Cartesian component [21]. In this work, we exploit the particle swarming nature of the approach to solve the orientation ambiguity by using the momentum of the agents. At each iteration, an individual agent is displaced according to the local principal stress vector *direction*. However, we select the displacement *orientation* that best aligns with the displacement of the agent in the previous iteration. In practice, using this approach, no sharp changes (i.e. larger than $90°$) in an agent's trajectory are possible.

Contrarily to intuition, the size of the agents steps does not depend on the magnitude of the stress vectors. To ensure that agents advance in a uniform front, every agent moves by a predefined and fixed step size. The uniformity of the front is a condition required to simplify the optimization problem, as we will see in the rest of this section. To ensure that no trajectories overlap (see Section III, Condition 1), we set the step size $h = l$, where l is the desired distance between neighboring agents.

To summarize, we consider an agent x_1 that at iteration k is located in $x_1(k)$ and the local principal stress vector $\mathbf{s}_1(k)$ (which we project on the slice if necessary). We indicate with $\hat{\mathbf{s}}_1(k) = {}^{\mathbf{s}_1(k)}/_{\|\mathbf{s}_1(k)\|_2}$ the normalized local principal stress vector. After one step, the agent will be located in

$$t_1(k+1) = x_1(k) + \mu\hat{\mathbf{s}}_1(k)h\,, \tag{2}$$

where μ is the momentum term used to circumvent the ambiguity problem: we set $\mu = 1$ when $\mathbf{s}_1(k) \cdot (x_1(k) - x_1(k-1)) \geq 0$ and $\mu = -1$ otherwise. $x_1(k-1)$ naturally denotes the location of the agent at the previous iteration of the algorithm, and $x_1(k) - x_1(k-1)$ corresponds to its last displacement. The forward step motion of two agents can be observed in Fig. 1.

2) Agents Potential Functions

Two potential functions are required to obtain the swarm potential introduced in Eq. (1): the environment potential P_e, and the aggregation potential P_a. The potential functions commonly utilized in the literature [38] are nonlinear, and the resulting total potential of a swarm is generally non-convex. Optimizing such functions remains a complex, inefficient, and time-consuming task [40], [41]. To simplify and accelerate the potential optimization problem, we introduce two quadratic environment and aggregation potential functions. Using these, Eq. (1) becomes a quadratic programming (QP) problem [42], which can be solved very efficiently by numerous existing solvers.

To define the environment potential P_e, let us consider an individual agent x_1 that has advanced by a step in the local stress direction following Eq. (2) and is now located in $t_1(k+1)$. We can imagine that $t_1(k+1)$ is the "ideal" location for the agent, as the trajectory of the agent between $x_1(k)$ and $t_1(k+1)$ is perfectly aligned with the local stress vector $\mathbf{s}_1(k)$. We assign the agent a *virtual mass* $m_1(k+1) \propto \|\mathbf{s}_1(k)\|_2$, that is proportional to the local stress vector magnitude[1]. We can now imagine the environment potential associated to the agent as the *energy* required to reposition the agent away from its ideal location $t_1(k+1)$, and bring it to a final location $x_1(k+1)$, as visible in Fig. 1. We define this as

$$P_e\big(x_1(k+1)\big) = m_1(k+1)\|x_1(k+1) - t_1(k+1)\|_2\,, \tag{3}$$

which is a quadratic function with a global minimum $P_e\big(x_1(k+1)\big) = 0$ when the agent is not repositioned. Using the virtual mass term m, we make the repositioning of agents with a larger associated stress more costly. This ensures that where the local stress is larger, the alignment between trajectories and stress will be higher.

The aggregation potential P_a is computed by comparing the locations of neighboring agents. As discussed in Section III, Condition 1, we want to avoid agent trajectories that overlap or cross. An intuitive way to achieve this would be to encode existing trajectories as locations to be avoided by the moving agents (for example, by including them in the environment potential). However, this approach presents a major scalability issue. As the number of iteration of Algorithm 1 grows, the size of existing trajectories keeps on increasing, making the size and complexity of the optimization problem larger and larger, and leading to computation issues. Additionally, this problem is amplified if the parts to be manufactured are large with respect to agent spacing. We avoid this issue altogether by ensuring that the moving agents never encounter

[1]In practice, we use the largest principal stress vector found in the FEA simulation as a normalization factor. We set, for each agent $x_i(k+1)$ and corresponding local stress vector $\mathbf{s}_i(k)$, a virtual mass $m_i(k+1) = {}^{\|\mathbf{s}_i(k)\|_2}/_{\max\|\mathbf{s}\|_2}$

any past trajectory. In this case, their potential functions can be designed to only depend on the location of other agents at the *current* iteration, and not at *all past* iterations. Instead of evaluating the distance between agents in a straight line, we decompose it into *radial distance*, which is measured perpendicularly to the displacement direction of the agents, and *axial distance*, measured in the displacement direction of the agents. We achieve an orderly advance of the agents by keeping neighboring agents at a desired radial distance and by minimizing their axial distance. Thus, the swarm advances as a front of homogeneously spaced agents, which are aligned along a quasi-straight front line. We consider two neighboring agents x_1 and x_2 whose locations at iteration $k+1$ are denoted as $x_1(k+1)$ and $x_2(k+1)$. We define the vectors

$$\mathbf{v} = x_2(k+1) - x_1(k+1)\,, \tag{4}$$

corresponding to the directed distance between the agents, and

$$\mathbf{d} = -\big(x_1(k) - x_1(k-1) + x_2(k) - x_2(k-1)\big)^{\perp}\,, \tag{5}$$

$$\hat{\mathbf{d}} = \frac{\mathbf{d}}{\|\mathbf{d}\|_2}\,, \tag{6}$$

the unit vector orthogonal[2] to the averaged direction of travel of the two agents in the previous iteration. Both vectors are shown in Fig. 1 for clarity. We define the aggregation potential P_a of agent x_1 with respect to its neighbor x_2 as

$$P_a\big(x_1(k+1)|_{x_2}\big) = (\|proj_{\hat{\mathbf{d}}}\mathbf{v}\|_2 - l)^2 + \|oproj_{\hat{\mathbf{d}}}\mathbf{v}\|_2^2\,, \tag{7}$$

where the norm in the first term, encoding the requirement to keep the radial distance close to the desired distance l, is computed as

$$\|proj_{\hat{\mathbf{d}}}\mathbf{v}\|_2 = \mathbf{v}\cdot\hat{\mathbf{d}}\,, \tag{8}$$

and the norm in the second term, used to minimize the axial distance, is computed as

$$\|oproj_{\hat{\mathbf{d}}}\mathbf{v}\|_2 = \|\mathbf{v} - (\mathbf{v}\cdot\hat{\mathbf{d}})\hat{\mathbf{d}}\|_2\,. \tag{9}$$

As $\hat{\mathbf{d}}$ and l are fixed, P_a is a quadratic function depending on $x_1(k+1)$ and $x_2(k+1)$, and has a global minimum $P_a\big(x_1(k+1)|_{x_2}\big) = 0$ when the two agents advance side by side at distance l.

3) Agents Adjacency

The aggregation potential P_a of the swarm is obtained by comparing the locations of different agents in the swarm. We greatly simplify the computation task by only comparing *neighboring* agents. Every agent has two neighbors, each being the closest agent in either orientation along the *radial* direction. In simpler terms, agent x_i has neighbors x_{i-1} and x_{i+1}; any other agent in the swarm has no aggregation potential with respect to x_i. The adjacency of agents is fixed since initialization and does not change during their motion through the part. Given the commutative nature of P_a as defined in Eq. (7), we consider as adjacent during the computations only agents x_i and x_{i+1}. This corresponds to having $P_a\big(x_i|_{x_{j\neq i+1}}\big) = 0$.

[2]The notation $\cdot^{\perp}$ used in Eq. (5) corresponds, in two dimensions, to a counterclockwise rotation of the vector by $90°$, i.e. $\mathbf{a}^{\perp} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\mathbf{a}$.

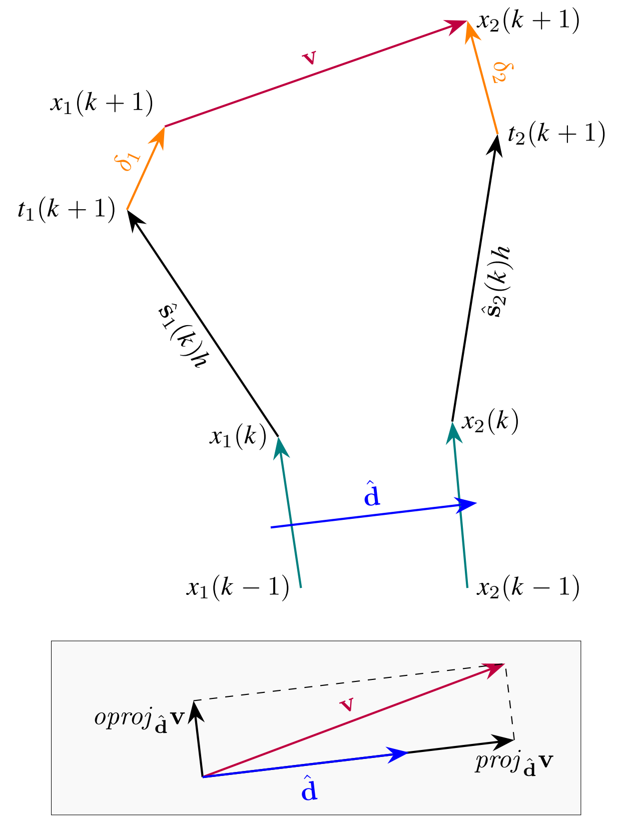


FIGURE 1. **Notation of the agents locations and distance vectors utilized in the definition of the environment potential P_e and of the aggregation potential P_a**

4) Boundary Conditions

To produce parts with a desired shape, the generated trajectories must cover the part entirely, as discussed in Section III, Condition 2. By simply defining a swarm of initial agents and making it propagate through the part as explained in Algorithm 1, complete coverage is not guaranteed. In particular, without further constraints, agents are free to step outside the part or to leave an empty non-covered space between the part boundaries and the swarm. To avoid such scenarios, we introduce *boundary conditions* to constrain and condition the swarm. As the swarm is initialized, at the beginning of Algorithm 1, two *boundary agents* are added at the extremities of the swarm. In a swarm $\mathcal{X}$ composed of N agents $x_{1,\ldots,N}$, we create the boundary agents x_0 and x_{N+1}, as shown in Fig. 2. The motion of these two agents is constrained to follow the part boundary. As a consequence, when computing the swarm trajectories, the agents x_1 and x_N maintain approximately a distance l from the part boundaries, ensuring complete coverage and avoiding boundaries crossing.

During trajectory generation, the swarm of agents can encounter other part boundaries (such as holes or other non-

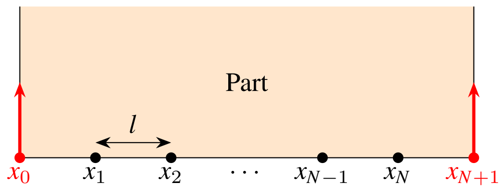


FIGURE 2. Initial swarm of equally spaced agents $x_{1,...,N}$ and boundary agents x_0 and x_{N+1}. The displacement of the boundary agents is one dimensional, as they are constrained to follow the part boundary.

convex features). In this case, two additional *internal* boundary agents are added to the swarm in between the two agents closest to the boundary, as shown in Fig. 3 The agents adjacency is updated accordingly, and the trajectory generation according to Algorithm 1 continues. Should the two internal boundary agents meet along their motion (for example, when the swarm has moved past a hole), they are removed from the swarm.

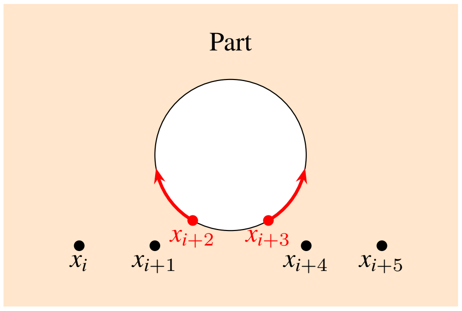


FIGURE 3. Internal boundary agents x_{i+2} and x_{i+3} added to the swarm when encountering a hole in the part

5) Optimization Problem

To compute the total potential of the swarm P, that we intend to minimize at each iteration, we first need to define the total environment and aggregation potentials of the swarm. Considering a swarm $\mathcal{X}$ of N agents $x_{1,...,N}$, at iteration $k+1$, the swarm environment potential is simply

$$P_e\big(\mathcal{X}(k+1)\big) = \sum_{i=1}^{N} P_e\big(x_i(k+1)\big)\,, \tag{10}$$

where the environment potential of an individual agent is given in Eq. (3). Similarly, we define the swarm aggregation potential as

$$P_a\big(\mathcal{X}(k+1)\big) = \sum_{i=0}^{N} P_a\big(x_i(k+1)|_{x_{i+1}}\big)\,, \tag{11}$$

where the aggregation potential of an individual agent is given in Eq. (7) and includes the notion of adjacency introduced in Section III-B3.

For a given tuning hyperparameter K, we can formulate the optimization problem being solved at each iteration of Algorithm 1 as

$$\begin{aligned}
\min_{x\in\mathcal{X}} \quad & P_a\big(\mathcal{X}(k+1)\big) + KP_e\big(\mathcal{X}(k+1)\big) && (12)\\
\text{s.t.} \quad & \|proj_{\hat{\mathbf{s}}_1}\delta_i(k+1)\|_2 \le h/4 \qquad i=1,\dots,N &&\\
& \|oproj_{\hat{\mathbf{s}}_1}\delta_i(k+1)\|_2 \le h/8 \qquad i=1,\dots,N &&\\
& x_i(k+1) \in \partial P \qquad \text{for boundary agents}\,, &&
\end{aligned}$$

where $\delta_i(k+1) = \big(x_1(k+1) - t_i(k+1)\big)$. The definitions of the projection terms in the constraints correspond to those given in Eq. (8) and Eq. (9), and ∂P indicates the boundary of the part. The inequality constraints limit the displacement of agents to a box around their ideal location $t_i(k+1)$, as shown in Fig. 4, and are necessary to ensure that agents keep advancing and do not cross. With a suitable change of coordinates, all constraints can be transformed into input bounds in the form $a_i \le x_i(k+1) \le b_i$. The resulting problem is a bound-constrained QP, which can be solved very efficiently.

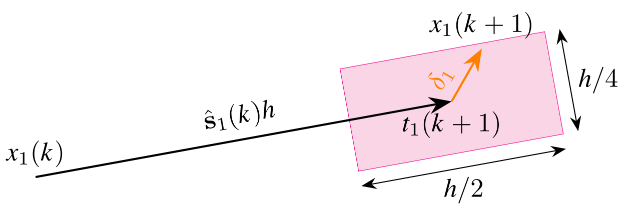


FIGURE 4. Representation of the constraints used in the optimization problem (12) to limit the displacement of the agents around their ideal location $t_i(k+1)$. The two constraints form the box depicted in magenta, which constrains $x_1(k+1)$.

6) Killing and Spawning

During trajectory generation, we exploit the fact that agents can be spawned or killed when necessary (see Section III, Condition 3). The procedure we utilize is based on the swarm potential P, following again the intuition that the ideal swarm behavior is achieved by minimizing P. At every iteration of Algorithm 1, we study the swarm (which we denote in this section with $\mathcal{X}_0(k+1)$) to identify the agent with the closest neighbor and the agent with the furthest neighbor. This corresponds to locating the regions where the trajectories are most compressed and most expanded. In the first case, we kill the *compressed* agent to form the swarm $\mathcal{X}_{-1}(k+1)$; in the second case, we spawn a new agent in the empty space between neighbors to form the swarm $\mathcal{X}_{+1}(k+1)$. We then solve the optimization problem (12) for $\mathcal{X}_0(k+1)$, $\mathcal{X}_{-1}(k+1)$, and $\mathcal{X}_{+1}(k+1)$. We denote as $\mathcal{P}\big(\mathcal{X}(k+1)\big)$ the solution of Eq. (12) for a swarm $\mathcal{X}(k+1)$. Finally, after computing the optimized potential of each swarm, and normalizing it with the number of agents in each swarm, we can make a comparison. We select and keep for the next iteration the swarm

$$\mathcal{X}(k+1) = \arg\min_{q=-1,0,1} \frac{\mathcal{P}\big(\mathcal{X}_q(k+1)\big)}{|\mathcal{X}_q|}\,, \tag{13}$$

which has the lowest potential among the evaluated scenarios. Clearly, this approach relies on heuristics which make only one spawning or one killing possible at each iteration. This choice was done to reduce the computational burden, as it allows Algorithm 1 to change the number of agents in the swarm in a principled way, while only solving Eq. (12) three times per iteration.

IV. RESULTS

In this section, we test and benchmark the swarm-based trajectory generation algorithm on a demonstrator part. The part we selected is a loaded open-hole tensile specimen. The specimen is taken from the ASTM standard for open-hole tensile strength of polymer matrix composite laminates [43], and described in Fig. 5. We first analyze the behavior of Algorithm 1 by studying the trajectories generated for the specimen. Then, we conduct an experiment to quantify the improvements obtained in the specimen strength via the proposed method.

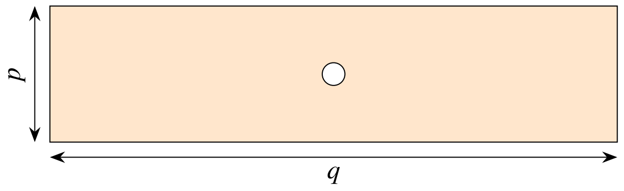


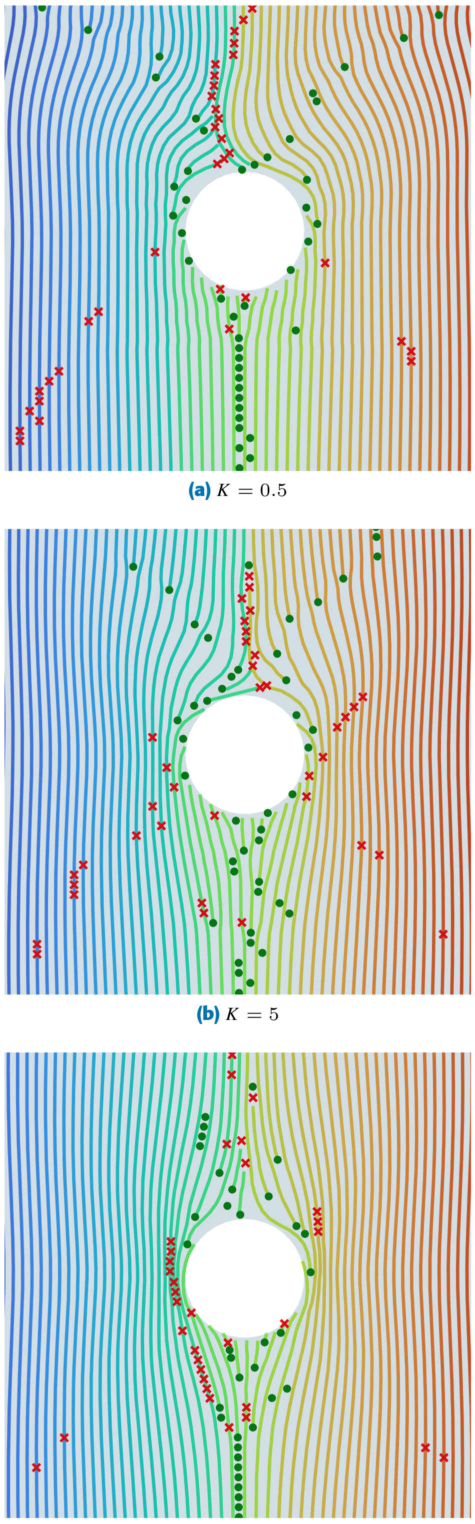
FIGURE 5. Open-hole tensile specimen according to [43]. The dimensions are $p = 36\,\mathrm{mm}$ and $q = 150\,\mathrm{mm}$, and the thickness is $2\,\mathrm{mm}$. The hole has a diameter of $6\,\mathrm{mm}$. The specimen is evaluated in a tensile strength test, with tension applied at the two narrow extremities.

A. TRAJECTORIES GENERATION

We implement the swarm-based trajectory generation algorithm in Python (including SciPy [44] and NumPy [45]); the optimization problems are solved efficiently by utilizing CasADi [46] as an interface and OSQP [47] as a solver. We first discuss the effect of the hyperparameter K, and then compare the trajectory generation performance with the method from [21].

1) Effect of K

In Eq. (12), K modifies the relative weight of the aggregation potential P_a with respect to the environment potential P_e. Selecting smaller values of K in the optimization produces swarms that mostly minimize P_e: this corresponds to better swarming behavior (i.e. agents advancing as a front and evenly spaced) at the expense of stress alignment. Conversely, larger values of K increase the importance of P_a in the optimization problem: the agents follow the stress in the part more, but the swarming behavior is worsened. In Fig. 6 we show the trajectories generated around the hole of the tensile specimen for three different values of K. With $K = 0.5$ (Fig. 6a) the trajectories are uniformly spaced, but they follow the stress circulating around the hole poorly, which reduces the manufactured part strength. With $K = 50$ (Fig. 6c) the



(a) $K = 0.5$

(b) $K = 5$

(c) $K = 50$

FIGURE 6. Comparison of manufacturing trajectories for the open-hole tensile specimen generated with different values of the hyperparameter *K*. The swarm advances from the bottom of the figures towards the top. Green circles and red crosses indicate the locations in which Algorithm 1 evaluates a potential spawn or kill, respectively (see Section III-B6).

trajectories follow the stress around the hole, but become very compressed in the regions to its left and right, which can cause defects in the manufacturing process. The intermediate case, produced with $K = 5$ (Fig. 6b) appears to constitute a satisfactory trade-off, with the trajectories showing a better stress following behavior than in the $K = 0.5$ case, and a better swarming behavior than in the $K = 50$ case.

To quantify the effect of K, we analyze stress alignment and trajectory spacing. For stress alignment, we introduce the metric

$$\bar{\beta} = \frac{\sum_{z=1}^{Z} m_z \|\hat{\mathbf{s}}_z \cdot \mathbf{p}_z\|}{\sum_{z=1}^{Z} m_z}, \tag{14}$$

where z indexes all the Z points in a set of generated trajectories, and $\hat{\mathbf{s}}_z$ and $\mathbf{p}_z$ are the normalized local stress and the printing direction at the point z. The alignment metric $\bar{\beta}$ is weighted by the magnitude of the local stress (encoded in the virtual mass m_z, see Section III-B2), and can range between 0 (no alignment) and 1 (perfect alignment). The alignment values for the three studied cases are given in Table 1. For trajectory spacing, we compute the distance of every point in the trajectory set from the following trajectory, and visualize the results as a distribution in Fig. 7. We also compute the variance of each distribution and report it in Table 1. Both metrics are in agreement with the qualitative intuitions obtained from Fig. 6, as both $\bar{\beta}$ and the variance of the distance distribution increase monotonically with K.

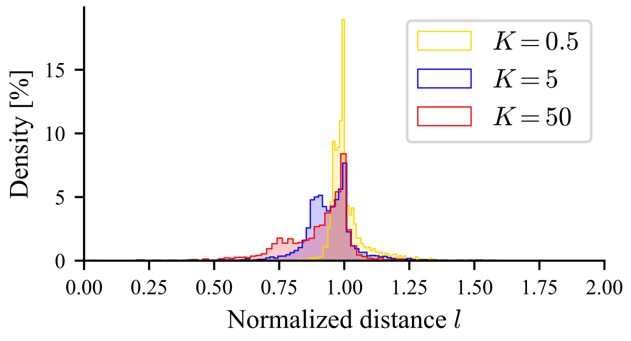


FIGURE 7. Distribution of distances between trajectories for different values of *K*. The distance has been normalized using the nominal line spacing *l*.

2) Comparison and Benchmarking

We compare Algorithm 1 with the method proposed in [21], a state-of-the-art method for stress-aligned trajectory generation for 3D printing, which we call *global trajectory generation*. We utilize both algorithms to generate print trajectories for one slice of the open-hole tensile specimen detailed in Fig. 5, with a nominal line spacing $l = 0.4\,\mathrm{mm}$. Both algorithms are executed on the same machine, a Windows computer with an Intel Core i9-9900K CPU running at $3.60\,\mathrm{GHz}$ and using $48\,\mathrm{GB}$ of RAM. The global trajectory generation algorithm solves a single large optimization problem and requires $22.7\,\mathrm{s}$ to produce the print trajectories. The proposed swarm-based trajectory generation algorithm solves iteratively a large number of simpler optimization problems and completes the task in $198\,\mathrm{ms}$. Thanks to its speed, our approach can be smoothly integrated in existing slicers without affecting user experience. The $99\,\%$ reduction in computation time is only one advantage of the method we propose in this work. As discussed by the authors in [21], the global trajectory generation method returns only one solution for a given part and load case, producing trajectories at a fixed (and quasi constant) distance. This is well suited to printing processes where the deposited line width cannot be changed, but penalizes stress alignment. In fact, the trajectories produced with the global method (also reported in Table 1) have $\bar{\beta} = 0.983$ and a variance of normalized distances of 4.4×10^{-4}. The trajectory spacing is extremely uniform, and the stress alignment is comparable with the results of Algorithm 1 with $K = 0.5$. When variable line width printing is possible, the swarm-based trajectory generation method offers more flexibility, as trajectories can be adapted by selecting the correct value of K. Allowing a certain amount of variation in the trajectories spacing increases stress alignment (as shown in Table 1) and as a consequence improves the mechanical properties of the part. The tuning of K is however limited by the hardware effectiveness in variable width printing. The stress alignment and trajectory spacing metrics we have proposed are a useful tool for choosing the most suitable value of K in practice. Based on these metrics, the ideal K can be found with numerous sampling based methods, such as for example random sampling, grid search, or Bayesian optimization.

B. PRINTED PARTS

We manufacture and test according to [43] 12 open-hole tensile specimen which are printed following different trajectories. The utilized trajectories are generated with:

1) A commercial slicer [48] set to *cross-hatching* infill. This infill creates a rectilinear grid by printing one layer as parallel lines in one direction, the next layer rotated by $90°$, etc. The direction of the lines is set to $45°$ with respect to the sides of the specimen. Lines are generated at a distance of $0.4\,\mathrm{mm}$ and with a $100\,\%$ infill. This case constitutes a baseline for non stress-aligned printing.
2) A commercial slicer [48] set to *aligned rectilinear* infill. All lines in the part are parallel and aligned in the direction of the long side of the specimen. Lines are generated at a distance of $0.4\,\mathrm{mm}$ and with a $100\,\%$ infill. This case constitutes a baseline for naive stress-aligned printing.
3) The global trajectory generation method from [21]. The nominal line distance is set to $0.4\,\mathrm{mm}$. This case is a state-of-the-art benchmark.
4) The proposed swarm-based trajectory generation method with $K = 5$ and a nominal line distance of $0.4\,\mathrm{mm}$. These print trajectories are shown in Fig. 8.

We manufacture three samples for each trajectory generation case. All parts are printed in PLA on a Prusa i3 MK3S

TABLE 1. Stress alignment and variance of the distribution of distances of the swarm-based trajectory generation method (for different values of *K*) and of the global trajectory generation method from [21]

	Swarm-based method			Global method from [21]
	$K = 0.5$	$K = 5$	$K = 50$	
$\bar{\beta}$	0.981	0.993	0.998	0.983
Variance	6.1×10^{-3}	12.9×10^{-3}	16.4×10^{-3}	4.4×10^{-4}

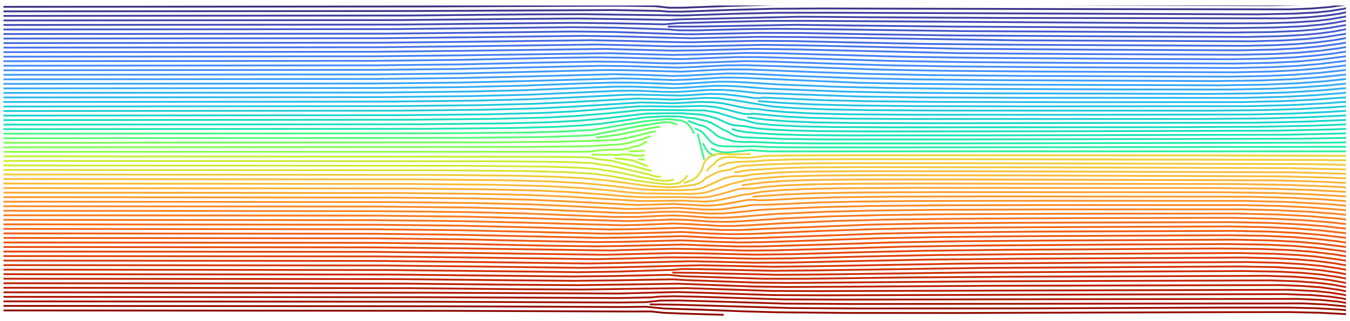


FIGURE 8. Print trajectories for the open-hole tensile specimen obtained with swarm-based trajectory generation (Algorithm 1) and $K = 5$. Agents move from the left to the right of the figure.

machine at a nozzle temperature of $205\,^{\circ}\mathrm{C}$, a bed temperature of $60\,^{\circ}\mathrm{C}$ and feed rate of $3150\,\mathrm{mm\,min^{-1}}$. Samples are tested on a Galdabini Quasar 10 testing machine following [43]. As different trajectory generation methods produce different line spacing distributions, and as this affects the deposition process, the samples have different densities. To make the results comparable, we divide the force measurements by the density of each sample to produce specific stress, specific modulus and specific strength (which are the density adjusted equivalents of stress, Young's modulus and ultimate tensile strength). The results are reported in Fig. 9 and Table 2. Cross-hatching, the most common trajectory generation technique in FFF, produces the worst performance, with the lowest specific modulus and strength. The aligned rectilinear approach and the global trajectory generation method have a very similar stress-strain curve, and a comparable specific modulus. The samples manufactured with our swarm-based trajectories outperform all other samples on the entire strain range. They produce a $\sim$ 10% improvement in specific modulus with respect to the aligned rectilinear and the global methods, while retaining the specific strength of the aligned rectilinear method. Contrarily to previous works [13], [21], this improvement was achieved on a conventional planar printer, using a ubiquitous feedstock material. It indicates that any combination of commercially available slicer and printer can benefit from our approach to produce stiffer and stronger parts via optimized trajectory generation. We point out that the high specific strength of the naive aligned rectilinear approach is in all likelihood the consequence of the line spacing homogeneity, which enables high quality deposition. We expect that the constant developments in the field of variable width FFF will reduce the number of deposition defects in the optimized parts, further increasing their strength. Furthermore, the aligned rectilinear approach can be used exclusively in very simple geometries where the load is rectilinear and stress-alignment can be achieved intuitively. Conversely, rigorous trajectory optimization methods can be applied to geometries and load cases of any complexity, as it was demonstrated in [21].

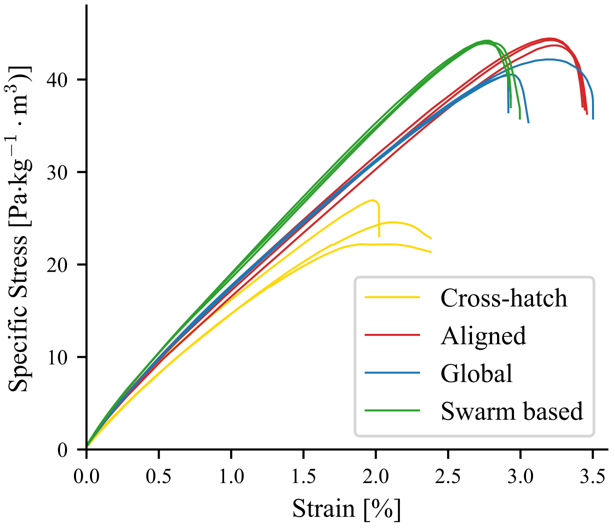


FIGURE 9. Specific stress-strain curve of the twelve tested open-hole tensile specimen

V. CONCLUSION

In this work, we introduced a novel swarm-based approach to the generation of optimized stress-aligned print trajectories

TABLE 2. Specific modulus and specific strength of the twelve tested open-hole tensile specimen. The values were obtained by averaging the results from samples produced with the same trajectory generation method.

	Cross-hatching	Aligned rectilinear	Global method from [21]	Swarm-based method ($K = 5$)
Specific modulus $[\mathrm{Pa\,kg^{-1}\,m^3}]$	17.48	19.53	19.91	**21.73**
Specific strength $[\mathrm{Pa\,kg^{-1}\,m^3}]$	24.57	44.14	41.08	**44.08**

for 3D printing. We evaluated our method and compared with state-of-the-art approaches, both in simulation and in experiments, focusing on the implications of stress-aligned trajectories for mechanical properties and printing quality. Our results demonstrate a significant improvement in computational efficiency with our proposed algorithm, achieving a remarkable $115\times$ increase in computation performance compared to the state-of-the-art method. This computational advantage, coupled with the flexibility of adapting trajectory spacing through hyperparameter tuning, positions our approach as a highly practical solution for seamless integration into existing slicers without compromising user experience. Furthermore, our method allows for enhanced stress alignment, leading to improved mechanical properties of printed parts. The specific modulus of parts produced using our swarm-based trajectories exhibited a $\sim$ 10% enhancement compared to existing methods. Our findings highlight the potential of trajectory optimization in advancing the mechanical performance and efficiency of 3D printing processes in general, and of the prevalent planar FFF of PLA in particular. In future research, we plan to extend the swarm-based approach to the non-planar slicing problem, and to solve the slicing and trajectory generation tasks in a single optimization.

ACKNOWLEDGMENT

We gratefully acknowledge the support of NematX AG which has provided the equipment to manufacture the samples.

REFERENCES

[1] "The state of 3d printing report: 2022 by sculpteo," https://www.sculpteo.com/en/ebooks/state-of-3d-printing-report-2022/, 2022, (Accessed on 01/04/2024).

[2] ISO/ASTM International, "Additive manufacturing — general principles — fundamentals and vocabulary," International Organization for Standardization, Standard ISO/ASTM 52900:2021, Nov. 2021.

[3] I. Gibson, D. Rosen, B. Stucker, M. Khorasani, D. Rosen, B. Stucker, and M. Khorasani, *Additive manufacturing technologies*. Springer, 2021, vol. 17.

[4] G. A. Nisja, A. Cao, and C. Gao, "Short review of nonplanar fused deposition modeling printing," *Material Design & Processing Communications*, vol. 3, no. 4, p. e221, 2021.

[5] D. Ahlers, F. Wasserfall, N. Hendrich, and J. Zhang, "3d printing of nonplanar layers for smooth surface generation," in *2019 IEEE 15th international conference on automation science and engineering (CASE)*. IEEE, 2019, pp. 1737–1743.

[6] M. J. Hooshmand, S. Mansour, and A. Dehghanian, "Optimization of build orientation in FFF using response surface methodology and posterior-based method," *Rapid Prototyping Journal*, vol. 27, no. 5, pp. 967–994, 2021.

[7] L. Di Angelo, P. Di Stefano, A. Dolatnezhadsomarin, E. Guardiani, and E. Khorram, "A reliable build orientation optimization method in additive manufacturing: The application to FDM technology," *The International Journal of Advanced Manufacturing Technology*, vol. 108, no. 1, pp. 263–276, 2020.

[8] P. Delfs, M. Tows, and H.-J. Schmid, "Optimized build orientation of additive manufactured parts for improved surface quality and build time," *Additive Manufacturing*, vol. 12, pp. 314–320, 2016.

[9] C. Wang and X. Qian, "Simultaneous optimization of build orientation and topology for additive manufacturing," *Additive Manufacturing*, vol. 34, p. 101246, 2020.

[10] B. Brenken, E. Barocio, A. Favaloro, V. Kunc, and R. B. Pipes, "Fused filament fabrication of fiber-reinforced polymers: A review," *Additive Manufacturing*, vol. 21, pp. 1–16, 2018.

[11] D. Jiang and D. E. Smith, "Anisotropic mechanical properties of oriented carbon fiber filled polymer composites produced with fused filament fabrication," *Additive Manufacturing*, vol. 18, pp. 84–94, 2017.

[12] L. Xia, S. Lin, and G. Ma, "Stress-based tool-path planning methodology for fused filament fabrication," *Additive Manufacturing*, vol. 32, p. 101020, 2020.

[13] G. Fang, T. Zhang, S. Zhong, X. Chen, Z. Zhong, and C. C. Wang, "Reinforced FDM: Multi-axis filament alignment with controlled anisotropic strength," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–15, 2020.

[14] X. Guidetti, A. Rupenyan, L. Fassl, M. Nabavi, and J. Lygeros, "Advanced manufacturing configuration by sample-efficient batch bayesian optimization," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 886–11 893, 2022.

[15] K. Barton, S. Kapila, and L. Y. L. Tse, "Fiber-reinforced 3d printing," Sep. 10 2019, uS Patent 10,406,750.

[16] X. Chen, G. Fang, W.-H. Liao, and C. C. Wang, "Field-based toolpath generation for 3d printing continuous fibre reinforced thermoplastic composites," *Additive Manufacturing*, vol. 49, p. 102470, 2022.

[17] Y. Yao, Y. Zhang, M. Aburaia, and M. Lackner, "3d printing of objects with continuous spatial paths by a multi-axis robotic FFF platform," *Applied Sciences*, vol. 11, no. 11, p. 4825, 2021.

[18] J. N. Reddy, *Introduction to the finite element method*. McGraw-Hill Education, 2019.

[19] A. F. Bower, *Applied mechanics of solids*. CRC press, 2009.

[20] S. Gantenbein, K. Masania, W. Woigk, J. P. Sesseg, T. A. Tervoort, and A. R. Studart, "Three-dimensional printing of hierarchical liquid-crystal-polymer structures," *Nature*, vol. 561, no. 7722, pp. 226–230, 2018.

[21] X. Guidetti, E. C. Balta, Y. Nagel, H. Yin, A. Rupenyan, and J. Lygeros, "Stress flow guided non-planar print trajectory optimization for additive manufacturing of anisotropic polymers," *Additive Manufacturing*, vol. 72, p. 103628, 2023.

[22] X. Guidetti, M. Kühne, Y. Nagel, E. C. Balta, A. Rupenyan, and J. Lygeros, "Data-driven process optimization of fused filament fabrication based on in situ measurements," *arXiv preprint arXiv:2210.15239*, 2022.

[23] G. Siqueira, D. Kokkinis, R. Libanori, M. K. Hausmann, A. S. Gladman, A. Neels, P. Tingaut, T. Zimmermann, J. A. Lewis, and A. R. Studart, "Cellulose nanocrystal inks for 3d printing of textured cellular architectures," *Advanced Functional Materials*, vol. 27, no. 12, p. 1604619, Feb. 2017.

[24] X. Guidetti, A. Mukne, M. Rueppel, Y. Nagel, E. C. Balta, and J. Lygeros, "Data-driven extrusion force control tuning for 3d printing," *arXiv preprint arXiv:2403.16470*, 2024.

[25] X. Guidetti, N. Mingard, R. Cruz-Oliver, Y. Nagel, M. Rueppel, A. Rupenyan, E. C. Balta, and J. Lygeros, "Force controlled printing for material extrusion additive manufacturing," *arXiv preprint arXiv:2403.16042*, 2024.

[26] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, pp. 1–41, 2013.

[27] L. Bayındır, "A review of swarm robotics tasks," *Neurocomputing*, vol. 172, pp. 292–321, 2016.

[28] S.-J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A survey on aerial swarm robotics," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 837–855, 2018.

[29] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[30] E. Rimon, *Exact robot navigation using artificial potential functions*. Yale University, 1990.

[31] E. Rimon and D. E. Kodischek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[32] J. H. Reif and H. Wang, "Social potential fields: A distributed behavioral control for autonomous robots," *Robotics and Autonomous Systems*, vol. 27, no. 3, pp. 171–194, 1999.

[33] K. Warburton and J. Lazarus, "Tendency-distance models of social cohesion in animal groups," *Journal of theoretical biology*, vol. 150, no. 4, pp. 473–488, 1991.

[34] D. Grünbaum and A. Okubo, "Modelling social animal aggregations," in *Frontiers in mathematical biology*. Springer, 1994, pp. 296–325.

[35] V. Gazi and K. M. Passino, "A class of attractions/repulsion functions for stable swarm aggregations," *International Journal of Control*, vol. 77, no. 18, pp. 1567–1579, 2004.

[36] ——, "Stability analysis of swarms," *IEEE transactions on automatic control*, vol. 48, no. 4, pp. 692–697, 2003.

[37] ——, "Stability analysis of social foraging swarms," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 539–557, 2004.

[38] V. Gazi, "On lagrangian dynamics based modeling of swarm behavior," *Physica D: Nonlinear Phenomena*, vol. 260, pp. 159–175, 2013.

[39] T. R. Metcalf, "Resolving the 180-degree ambiguity in vector magnetic field measurements: The 'minimum'energy solution," *Solar Physics*, vol. 155, pp. 235–242, 1994.

[40] M. Danilova, P. Dvurechensky, A. Gasnikov, E. Gorbunov, S. Guminov, D. Kamzolov, and I. Shibaev, "Recent theoretical advances in non-convex optimization," in *High-Dimensional Optimization and Probability: With a View Towards Data Science*. Springer, 2022, pp. 79–163.

[41] J. A. Snyman, D. N. Wilke *et al.*, *Practical mathematical optimization*. Springer, 2005.

[42] M. Frank, P. Wolfe *et al.*, "An algorithm for quadratic programming," *Naval research logistics quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[43] "Standard test method for open-hole tensile strength of polymer matrix composite laminates," American Society for Testing and Materials, Tech. Rep. D5766/D5766M – 23, 2023.

[44] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[45] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[46] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, 2018.

[47] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020. [Online]. Available: https://doi.org/10.1007/s12532-020-00179-2

[48] "Prusaslicer 2.7.2," https://www.prusa3d.com/en/page/prusaslicer_424/, 2024, (Released on 29/02/2024).