Solid Freeform Fabrication 2017: Proceedings of the 28th Annual International Solid Freeform Fabrication Symposium – An Additive Manufacturing Conference
Reviewed Paper

TIME-OPTIMAL SCAN PATH PLANNING BASED ON ANALYSIS OF SLICED GEOMETRY

Yi Xiong*, Anke Van Campen*, Anje Van Vlierberghe*,
Karolien Kempen†, and Jean-Pierre Kruth†

* Flanders Make vzw, 3001 Heverlee, Belgium
† Division PMA, Department of Mechanical Engineering, Katholieke University Leuven, 3001 Heverlee, Belgium

Abstract

In powder-bed based layered manufacturing, a focused and high power laser beam is guided to travel through pre-defined trajectories with various process parameters such as the scan speed, laser power, and beam diameter to consolidate powdered materials together. The pre-defined path, therefore, plays a significant role not only on the build part quality but also on the build time. Current path planning strategies are defined only on the layer level. Although, contours on one layer can be significantly different in the geometry shape. This paper proposes an adaptive scan path planning method based on the geometric characteristic of contours. With this approach, the user is able to control and optimize the scan path for contours with different geometric types. An algorithm for determining the scanning direction to minimize the build time is discussed in detail. A path planning approach for non-productive paths illustrates the potential time gain applying more intelligent strategies.

1. Introduction

Powder-bed based layered manufacturing is a widely adopted additive manufacturing approach for rapid prototyping and for direct manufacturing of parts. It is one of the seven major additive manufacturing processes categorized in the ASTM standard [1]. The process uses a focused and high power laser beam to consolidate a wide range of powdered materials such as metals, polymers, ceramics, and other composites [2].

The process planning in the powder-bed layered manufacturing process is similar to the typical additive manufacturing pipeline, which is shown in Figure 1. The three-dimensional (3D) virtual part designed within computer-aided design (CAD) software is first sliced into a stack of two-dimensional (2D) slices at small intervals, which are typically about tens of μm. File formats, for example the Common Layer Interface (CLI) and Slice Layer Interface (SLI), are used to store the sliced geometry with polygon based boundary representations. Then, 2D scan paths will be generated in the path planning (also called hatching) step to fill these closed boundary geometries on each slice along the build direction. During the manufacturing process, the laser beam is deflected by galvanometer scanners to conduct pre-defined motions on a 2D plane and to fuse powdered materials on the scan path. This consolidation process is repeated for each slice to finally form the 3D CAD part.

Numbers of decision making processes are involved at different steps in the workflow such as the selection of slicing orientation, hatching parameters, and scanning parameters etc. In the path planning step, the final scan path varies depending on parameter selection including pattern, direction, and line space, but also it limits the process and machine dynamics. These parameters have fundamental influence on the process at different levels going from the melt pool, via the scan track, until the final build part. An optimal parameter set is obtained through

extensive experiments for producing parts with high performance e.g., high density, less residual deformation, and accurate geometry. It is obvious that the selection of process parameters is pivotal for both the build part quality and the time and material costs.

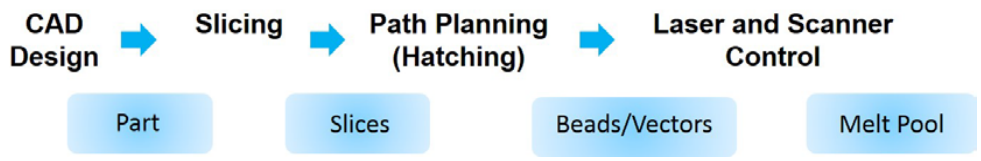


Figure 1. *The workflow of process planning for the powder-bed based layered manufacturing process.*

Selecting the process parameters can be conducted at three levels: part level, slice level, and vector level. At the part level, various optimization tools for selecting the slice orientation to minimize production time and roughness on a specific surface are already available in commercially available software. At the vector level, the approach to obtain processing windows for selecting scanning parameters has been intensively studied [3]. At the slice level, previous endeavors have studied the influence of different scan path patterns on the build part quality [4, 5]. However, the opportunity to shorten the build time of powder-bed based layered manufacturing through the path z has not yet been fully explored. Therefore, this paper aims at identifying different possibilities to reduce the build time and propose the corresponding path planning at the slice level.

2. Background

Build time modelling

Optimization of the build time inherent to the scan path requires a proper process model and the selection of a path planning strategy. State-of-practice relies on models estimating the build time either based on volumetric calculations, the number of layers, or the length of all scan vectors. However, these models are an excessively simplified representation of the actual process, and do not take into account various delays which are enforced due to the system dynamics. To control the motion of the scanner, a series of scan vectors are generated in the path planning step. The basic motion of the laser beam is mostly linear. The laser beam travels from a start position to a target position defined by the scan vector. Depending on whether the laser is on or off while travelling, the scan vectors are classified as mark (productive) vectors or jump (non-productive), respectively. Delayed responses exist between sending the command and the actual movement or laser on/off takes place (overview of possible delays and their sources are listed in Table 1). This is due to the dynamic behaviors of different system components, i.e., the laser and scanner. To ensure target positions are always reached, a start and stop motion is widely adopted in powder-bed based machines [6].

The laser power and scan speed are parameters that need to be controlled in a synchronized way so that a desired energy input is guaranteed (the ratio between the laser power/ scan speed). The laser power is regulated at different speeds. The controller, therefore, has to take into account the aforementioned response delays in the subsystem. Different delays (waiting times) are assigned in the start and stop motion for both the laser and the scanner. There are a total of five delays where two are related to the laser and three are related to the scanner, as summarized in Table 1. The laser delay time does not affect the total build time but has an influence on the

build part quality. Due to the fact that the jump speed is faster than the mark speed, which excites the system more, the jump delay is typically larger than the mark delay.

Table 1. *Different delays used in the powder-bed based layered manufacturing and their typical values*

	Delay types	Typical values [7]	Description
Laser	Laser on delay d_{on}	100 µs	The delay before the laser is turned on
	Laser off delay d_{off}	100 µs	The delay before the laser is turned off
Scanner	Jump delay d_{jump}	250 µs	The delay after a jump vector
	Mark delay d_{mark}	100 µs	The delay between a mark vector and a jump vector
	Polygon delay d_{poly}	50 µs	The delay between two mark vectors

To illustrate the different delays, an example for scanning the letters 'FM' is shown in Figure 2 (a). Delays for the start command and the end command are the same in most scenarios. Therefore, the total build time, as shown in Figure 2 (b), can be divided into two parts: the motion part (marked in a dotted pattern) and the waiting part (marked in a diagonal stripes pattern).

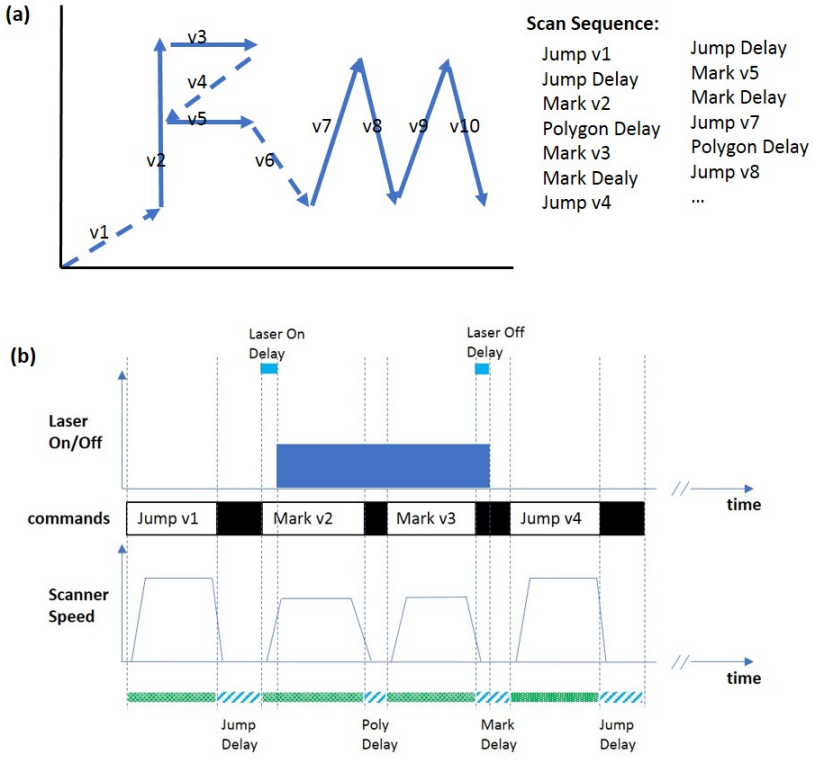


Figure 2. *(a) A series of scan vectors and delays for scanning the letters 'FM';(b) timing of the laser and scanner for executing scan vectors 1 to 4.*

The total build time T_{total} does not only depend on the scanning path but is also constrained by the limits of the process and machine. It can then be modelled as:

$$T_{\text{total}} = T_{\text{mark}} + T_{\text{jump}} + T_{\text{delay}}, \quad (1)$$

where T_{mark} is the mark time, T_{jump} is the jump time, and T_{delay} is the delay time for three scanner related delays. The mark time T_{mark} can be calculated as:

$$T_{\text{mark}} = \sum_{m=1}^{\text{M}} t_m^{\text{mark}} = \sum_{m=1}^{\text{M}} (t_m^{\text{acc}} + t_m^{\text{cns}} + t_m^{\text{dcc}}), \quad (2)$$

where M is the number of mark vector, t_m^{acc}, t_m^{cns}, and t_m^{dcc} are delays for the acceleration, constant speed motion, and deceleration stages, respectively. However, it should be noted that depending on the mark vector and the process and machine limits, there will be only acceleration and deceleration stages. T_{jump} can be calculated as:

$$T_{\text{jump}} = \sum_{j=1}^{\text{J}} t_j^{\text{jump}} = \sum_{j=1}^{\text{J}} (t_j^{\text{acc}} + t_j^{\text{cns}} + t_j^{\text{dcc}}), \quad (3)$$

where J is the number of jump vector, t_j^{acc}, t_j^{cns}, and t_j^{dcc} are time costs for the acceleration, constant speed motion, and deceleration, respectively. In addition, T_{delay} can be calculated as:

$$T_{\text{dealy}} = \sum_{n=1}^{\text{N}} t_n^{\text{delay}} = n_{\text{j}} d_{\text{jump}} + n_{\text{m}} d_{\text{mark}} + n_{\text{p}} d_{\text{poly}}, \quad (4)$$

where n_{j}, n_{m}, and n_{p} are numbers of jump delay, mark delay, and polygon delay, respectively. In addition, d_{jump}, d_{mark}, and d_{poly} correspond to the jump delay, mark delay, and polygon delays.

Reflection on time gain possibilities

There are several possibilities to reduce the build time using the powder-bed based layer manufacturing. These opportunities can be summarized from three aspects:

1. Reduction of the number of mark and jump vectors. The lower the number of vectors, the lower the total time delay as indicated in Equation 4. Related to mark vectors, for sliced geometry with a clear directionality, the selection of the optimal direction effectively reduces the number of vectors as will be shown in more detail in section 3. Related to jump vectors, the number of vectors can be reduced to a minimum by avoiding unnecessary vectors.

2. Planning the trajectory of mark and jump vectors so that the travel time is minimized, as will be illustrated in Section 4, for jump vectors connecting layer parts.

3. Dealing with machine dynamics so that delay times can be reduced. To obtain a faster and accurate response of the machine, feedforward control techniques such as input shaping [8] can be applied. Also, path following techniques are worthwhile studying [9], where special attention should be paid to the compromise between geometrical accuracy, part quality and the process time gain.

3. Selection of Scanning Direction

For some scanning patterns, especially zig-zag and parallel line patterns, the selection of scanning direction has a significant influence on the number of scan vectors. Optimizing the scanning direction is, therefore, an effective way to reduce the number of scan vectors and naturally reduce the build time. However, scanning parameters are usually defined on the layer level in current commercial software. The end-user can only select one scanning direction or patterns with fixed directions for all the contours within one layer despite the variance in their geometric shapes. For example, to scan a rectangular contour, the number of scan vectors for scanning along the long axis can be significantly smaller than the number of scan vectors for scanning along the short axis. Previous works [4, 10] attempted to determine the scanning direction with a minimized number of scan vectors through exhaustive search algorithms, which are computationally expensive. It is necessary to propose an algorithm that gives a minimized number of scan vectors based on the geometric analysis of the polygon contours. The objective of the algorithm presented here can be summarized as: finding a direction that minimizes the number of scan vectors based on the geometric properties.

It is found in [11] that the scanning direction, which generates a minimal number of passes to fill a convex polygon, must be parallel to one of its edges. Using this finding, an algorithm is proposed based on the calculation of the distance from all the vertices to every edge of the polygon. However, this algorithm can only derive the scanning direction with minimum number of scan vectors, but it cannot detect the influence of the scanning direction selection on the build time. To tackle this, the following assumption has been made: for a given polygon, the number of parallel lines to fill/hatch the polygon is inversely proportional to the sum of the projection of all polygon edges in the scanning direction.

The problem with finding a direction to minimize the number of scan vectors can be restated as finding a direction that has a minimum sum of the projection of polygon edges in that direction. As shown in Figure 3, the build time is estimated for each scanning direction using an exhaustive search algorithm. Meanwhile, the sum of edge projections is also calculated. The comparison shows a close correlation between the proposed indicator and the build time, which verifies the assumption. Additionally, more scan vectors with longer lengths are generated within the time minimization scanning direction than other directions. However, one should note that the long scan vector is not always desired, since it can cause possible negative effects e.g., large residual deformations [12]. Therefore, the scanning direction, which results in superior overall performance, is selected in practice instead of the time minimization scanning direction. Yet, the rest of this section will take the time minimized direction as an example to show the potential of reducing the build time through the scanning direction selection.

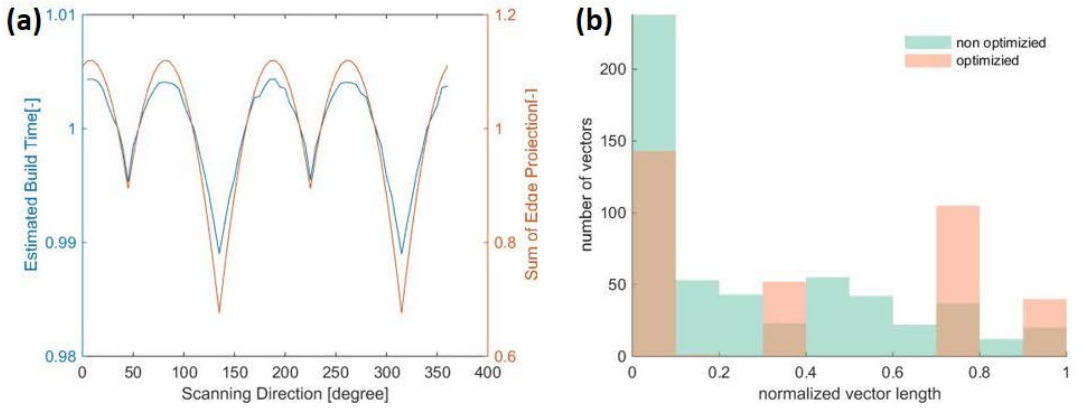


Figure 3. *(a) The build time versus the sum of edge projections in different scanning directions; (b) the histogram of the vector length for the scan vectors generated with the non-optimized scanning direction and with the time-optimal scanning direction.*

To find the time minimized scanning direction, based on the findings in [11] and aforementioned assumption, the algorithm is formulated as:

1. Create a list of the orientation of all polygon edges;
2. Calculate the sum of the projection of polygon edges for all directions within the previous list;
3. Find the minimum sum of the projection of polygon edges. The corresponding direction is the time minimized scanning direction.

One example of the algorithm implementation is presented in Figure 4.

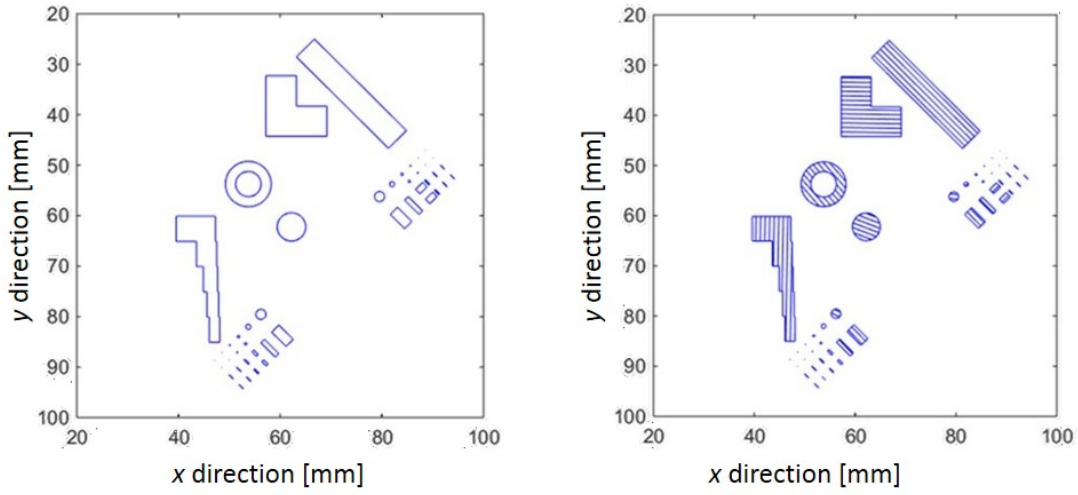
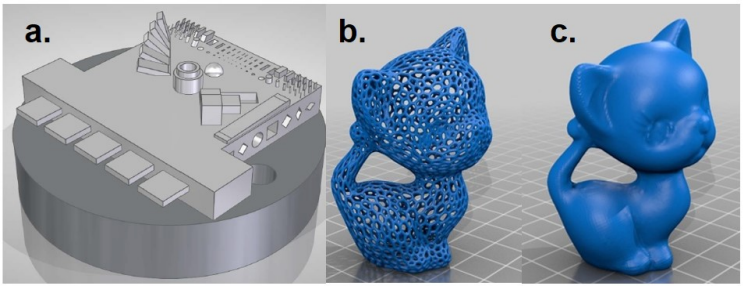


Figure 4. *Contours on one layer hatched with optimal directions calculated based on the proposed algorithm.*

To check the performance of the proposed algorithm, three different parts have been used to compare the build time with and without the time minimized scanning paths based on the time-cost model introduced in Section 2. The performance with different models is shown in

Figure 5. It can be found that for a test artifact, the time gain is about 7.6 %. However, for a Voronoi Kitten the time saved is less. For a solid Kitten, the time saved is negligible, and it is mainly due to the fact that sliced geometries are mostly circular shapes. It supports the fact that the proposed algorithm saves more time for sliced geometries with a clear directionality.



Part #	One Fixed Angle	Adaptive Hatching
a	76.81 min	70.99 min (~7.6%)
b	26.17 min	25.54 min (~2.4%)
c	52.10 min	52.08 min (~0.04%)

Figure 5. *Models used for build time simulations: (a) a test artifact; (b) a Voronoi kitten; and (c) a solid kitten. The build time is compared between using one fixed scanning direction and using the time minimization direction.*

4. Minimization of jump time between layer parts

In this section, we focus on the path planning of the smallest set of jump vectors between layer parts. To the authors' knowledge, not much has been reported related to the minimization of the non-productive time i.e. jump time in AM processes. De Wil et al. [13] provide a recent overview and classification of path planning algorithms albeit for cutting problems. Livesu et al. [14] refer in their review to two studies: one applying a heuristic approach and the other applying a metaheuristic approach. Unfortunately, no details on the kinematic model are disclosed nor are machine and process parameters. In general, the path planning problem for the jump vectors can be formulated as an integer problem given the nature of its variables being the numbering assigned to each layer part $s \in \mathbb{Z}$. The current industrial state-of-practice in AM still relies on path planning approaches which might be random decision making or moving to the nearest neighbor i.e. the shortest distance between layer parts. Applying the former, the path can be accidently time optimal or the time worst case. Applying the latter, there is no guarantee that jump time is minimized due to the kinematics involved.

The algorithms should provide their solutions within an acceptable computational time. Mainly four aspects influence the path planning when aiming for time minimization:

- the number of layer parts e.g. lattice structures;
- the location of the layer parts relative to each other;
- the direction of the last mark vector before the jump;
- the machine and process kinematics.

One can imagine that when the number of layer parts grow, the complexity of the path planning algorithm grows as well as the computational time (more than exponentially), which is not desirable. To tackle the path planning for the jump vectors, we made some considerations as listed in Table 2.

Table 2. *Considerations on path planning approaches: + or - indicates the evaluation is or is not in favor of the approach. P(x*) indicates the chance to reach the global optimum.*

	Nearest Neighbor	**Metaheuristics**	**Dynamic Programming**	**Integer programming**
Solver-dependency	+++	--	+++	---
Scalability	++	---	-	--
Modelling (physical) rules	--	--	+++	++
Computational cost	+++	---	+	+
P(x*)	---	+	++	++

Based on these considerations, we opted to use dynamic programming (DP). There is no solver-dependency and, therefore, no extra costs; the possibility to introduce constraints and rules (partly) compensates for the poorer scalability and the computational cost, and it enhances the chance to at least approach the global optimum i.e. the overall most time saving path. We will benchmark our algorithm in terms of time gained to a nearest neighbor (NN) approach. Both are custom implementations in MATLAB [15].

In our proof-of-concept case, all layer parts S are allowed to be the starting point. The scanner can jump to any layer part except itself and the ones already visited. Hence, the number of solutions to a given problem with S layer parts grows factorial, e.g. if S = 5, the number of solutions will be 5! = 120. This number can considerably be reduced by including rules. An obvious rule is that within an iteration a layer part is the next step in multiple paths, only the fastest among these paths is kept as the others are excluded from further evaluation. Other examples of potential rules are: restricting the number of paths to be evaluated depending on the iteration whenever the problem size becomes too large risking to lose global optimality in favor of computational cost, restricting start and/or end points, adding thermo-dynamical constraints, etc.

To calculate the jump time, kinematics are modelled as follows:

$$\dot{\boldsymbol{x}}_{j,0} \le \dot{\boldsymbol{x}}_{\max}^{\text{process}}, \left|\dot{\boldsymbol{x}}_{j,0}\right| = \left|\dot{\boldsymbol{x}}_{\max}^{process}\right|, \dot{\boldsymbol{x}}_{j,\max} \le \dot{\boldsymbol{x}}_{\max}^{\text{machine}}, \dot{\boldsymbol{x}}_{j,\text{end}} = 0, \ddot{\boldsymbol{x}}_j = \ddot{\boldsymbol{x}}_{\max}^{\text{machine}} \tag{5}$$

$$\left.\begin{aligned} t^{\text{dcc},1} &= \dot{\mathbf{x}}_{j,0} \big/ \ddot{\mathbf{x}}_j , \\ t^{\text{dcc},2} &= (\dot{\mathbf{x}}_j^* - \dot{\mathbf{x}}_{j,0}) \big/ \ddot{\mathbf{x}}_j , \\ t^{\text{dcc},4} &= \dot{\mathbf{x}}_j^* \big/ \ddot{\mathbf{x}}_j , \end{aligned}\right\} \quad t_{s\to s+1} = t^{\text{dcc},1} + t^{\text{acc},2} + t^{\text{cns},3} + t^{\text{dcc},4} \tag{6}$$

$$\dot{\boldsymbol{x}}_j^* = \max \dot{\boldsymbol{x}}_j \tag{7}$$

subject to

$$\mathbf{x}_{s+1} = \mathbf{x}_j^{\text{dcc},1} + \mathbf{x}_j^{\text{acc},2} + \mathbf{x}_j^{\text{cns},3} + \mathbf{x}_j^{\text{dcc},4}$$

$$\boldsymbol{x}_j^{\mathrm{dcc},1} = \ddot{\boldsymbol{x}}_j (t^{\mathrm{dcc},1})^2 / 2$$
$$\mathbf{x}_j^{\mathrm{cte},3} = \dot{\mathbf{x}}_j^* t^{\mathrm{cns},3}$$
$$\boldsymbol{x}_j^{\mathrm{dcc},4} = \dot{\boldsymbol{x}}_j^* t^{\mathrm{dcc},4} + \ddot{\boldsymbol{x}}_j (t^{\mathrm{dcc},4})^2 / 2$$

where $t^{\mathrm{dcc/acc/cns}}$ is the deceleration/acceleration/constant velocity time in the respective segment (see also Figure 6), $t_{s\to s+1}$ is the jump travel time from layer part *s* to the next layer part *s+1* at distance $\boldsymbol{x}$, $(.)_{\max}^{\mathrm{process/machine}}$ are the process/machine limits for velocity $\dot{\boldsymbol{x}}$ and acceleration $\ddot{\boldsymbol{x}}$, kinematics at the start/end of the jump *j* are indicated with subscript 0/end, and $\dot{\boldsymbol{x}}_j^*$ are the maximum velocity (lower than machine limits) during jump *j*. $t^{\mathrm{dcc},1}$ is zero if $|\boldsymbol{x}_j^{\mathrm{dcc},1}|$ is smaller than $|\boldsymbol{x}_{s+1}|$ in the respective direction. $t^{\mathrm{acc},2/\mathrm{cns},3/\mathrm{dcc},4}$ is zero if $|\boldsymbol{x}_j^{\mathrm{dcc},1}|$ equals $|\boldsymbol{x}_{s+1}|$ in the respective direction. $t^{\mathrm{cns},3}$ is zero if $\dot{\boldsymbol{x}}_j^* < \dot{\boldsymbol{x}}_{\max}^{\mathrm{machine}}$. Figure 6 illustrates three different scenarios related to our case, hence these are scenarios for the Y direction.

In our simulation model, machine limits were set to 2 m/s and 20 m/s^2, process limits were set to 0.3 m/s and 2 m/s^2. The velocity in X was always 0 m/s and the velocity in Y equaled always the process limit, hence 0.3 m/s at the start of the jump. We assumed that a certain layer with an area of 1 cm^2 contained a number of layer parts S, which should be processed sequentially. The contour was scanned first before the inner region, and hatching started and ended at known locations. The hatching end point was the take off point for the jump vector.

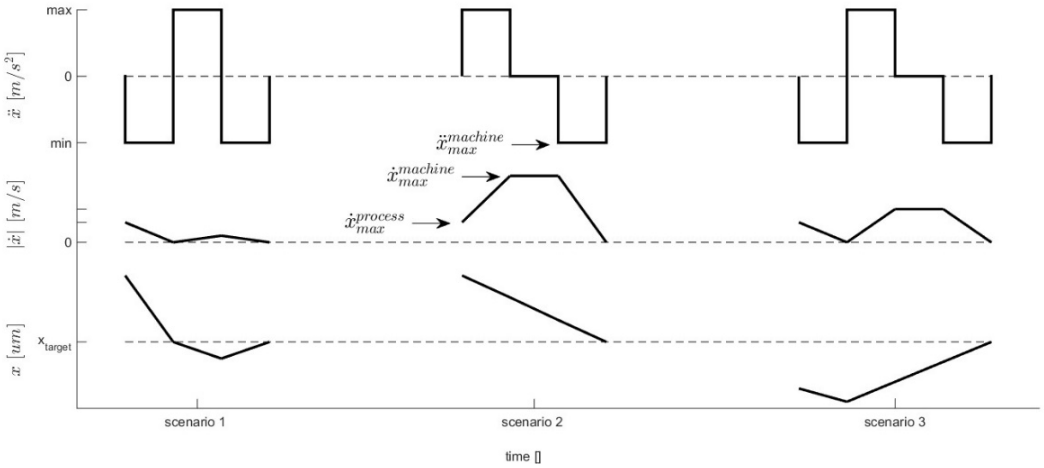


Figure 6. *<u>Scenario 1:</u> the scanner immediately decelerates from the jump start. The distance to the next layer part is too small, hence the passes the target point decelerating to stand still and moves back. <u>Scenario 2:</u> the distance to the next layer part allows acceleration, here the scanner reaches the machine limit and decelerates to stand still at the target point. <u>Scenario 3:</u> due to direction change, the scanner needs to decelerate to stand still and will than move to the target point in the opposite direction. Note that the accelerations are a theoretical representation. Indeed, the change in acceleration becomes infinite when switching directions, which is obviously not desirable.*

The point on the contour of the next layer part closest to the hatching start point of the respective layer part was the end point of the jump vector. Therefore, a dummy mark vector was inserted in case this point was not the beginning nor the end of a mark vector. This is illustrated in Figure 7. For now, we restricted the path planning, and the time minimization to

jump vectors between layer parts. As a consequence, the minimum set of jump vectors is J = S-1. The effect on the total process time of the dummy vector is not relevant in light of the benefits related to applying (intelligent) path planning for the jump vectors as will be further presented here. Regarding the NN algorithm, we did not enforce a constraint on the start point any layer part can be the start point. For example, in case of five layer parts, five NN paths are calculated. At the end, the fastest NN path is selected for comparison to the fastest DP path.

Figure 7. *Illustration of a jump vector (dotted line) from layer part A to layer part B. The black dots indicate the starts and ends of the mark vectors representing the contour. The grey arrows indicate the hatching lines. The grey rectangle/square represent the start/end of the first/last hatching line. The square is also the take-off point of the jump vector. The end point of the jump vector is the black cross on the contour of layer part B.*

Results of our simulation study are listed in Table 3 for 5, 20, 50, and 200 layer parts. The absolute simulated travelling times are shown, as well as the differences between NN and DP relative to the DP time. Figure 8 visualizes the position of the layer parts.

Table 3. *Comparison between potential time gains when adopting more advanced path planning, here DP, instead of a nearest neighbor approach. These simulated jump times are for one layer and a (sampled) lattice structure of 1 cm^2. * indicates the solution is globally optimal and it has been verified within the complete solution space, ‘ indicates the solution is locally optimal.*

	DP time [s]	NN time [s]	Time difference relative [%]	Time difference absolute [s]
S = 5*	.165	.2	10%	.035
S = 20*	.74	.89	21%	.150
S’ = 50	1.94	2.24	15%	.300
S’ = 200	8.01	9.24	15%	1.23

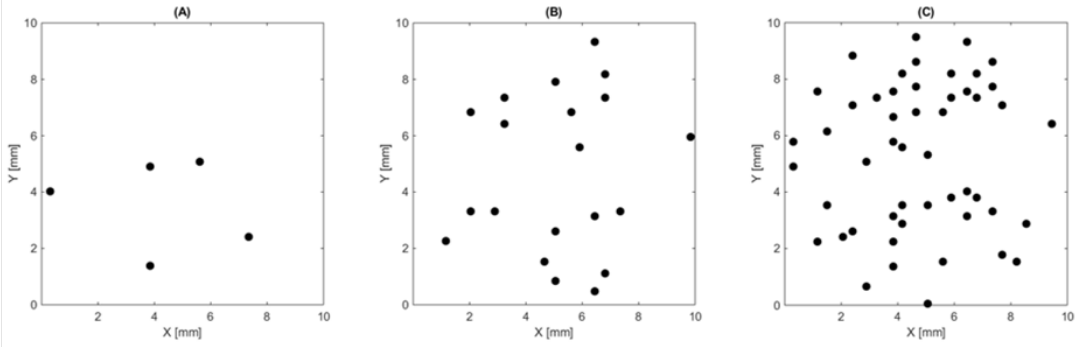


Figure 8. *Illustration of the location of the different layer parts in the layer as subsampled from 200 subparts: (A) 5 layer parts, (B) 20 layer parts, (C) 50 layer parts.*

Our results show that even for a limited number of layer parts, adopting a DP approach is worthwhile in light of time gain. To illustrate the influence of kinematic parameters, when adopting a process limit of 0.6 m/s, our DP algorithm outperformed our NN approach by almost 2 seconds for the case of 200 layer parts.

Altogether, we showed the potential of applying intelligent path planning strategies to reduce jump time between layer parts. Obviously, path planning should not be a constraint to the jump vectors, also mark vectors should be considered. The main challenge will be to keep the computational cost acceptable by defining an appropriate set of constraints and rules.

5. Conclusions

In this paper, we presented a general framework for reducing the manufacturing time in the laser powder bed fusion process. A build time model, which considers the dynamics of the scan system is given for the process. Opportunities for reducing the build time have been identified. Different examples have been presented for reducing the number of jump vectors through selecting a time optimal scanning direction and for reducing the jump time between layer parts using dynamic programming. Improvements regarding the build time has been observed for both algorithms. Meanwhile, one should note that the time gain for both algorithms are product configuration dependent. For the time optimal scan direction algorithm, most improvements are gained with sliced geometries that have a clear directionality. For the dynamic programming algorithm, the scan layer with multiple layer parts, e.g., scaffold/ cellular parts results in the largest absolute time gains.

In future work, optimization of the scan path on other aspects, e.g., the residual deformation, the accuracy will also be considered. A multi-dimensional optimization framework will be composed. Regarding time minimization in the AM process, we will also focus on reducing the delays after the jump and mark vectors

Acknowledgment

The research leading to these results was performed within the Combilaser SBO project and received funding from VLAIO (Flemish government agency for Innovation and Entrepreneurship) under the grant agreement No. IWT.150521.

References

[1] Gibson I, Rosen DW, Stucker B. *Additive Manufacturing Technologies*. New York: Springer, 2010.

[2] Kruth JP, Levy G, Klocke F, et al. Consolidation phenomena in laser and powder-bed based layered manufacturing. *CIRP Ann - Manuf Technol* 2007; 56: 730–759.

[3] Yadroitsev I, Bertrand P, Smurov I. Parametric analysis of the selective laser melting process. *Appl Surf Sci* 2007; 253: 8064–8069.

[4] Jin Y an, He Y, Fu J zhong, et al. Optimization of tool-path generation for material extrusion-based additive manufacturing technology. *Addit Manuf* 2014; 1: 32–47.

[5] Qian B, Shi YS, Wei QS, et al. The helix scan strategy applied to the selective laser melting. *Int J Adv Manuf Technol* 2012; 63: 631–640.

[6] Yeung H, Neira J, Lane B, et al. Laser path planning and power control strategies for powder bed fusion systems. In: *The Solid Freeform Fabrication Symposium*. Austin,Texas, 2016, pp. 113–127.

[7] Scanlabs. Installation and Operation The RTC ® 5 PC Interface Board and RTC ® 5 PC / 104- Plus Board. 2010; 1–467.
[8] Buls S, Craeghs T, Clijsters S, et al. The influence of a dynamically optimized galvano based laser scanner on the total scan time of slm parts. In: *The Solid Freeform Fabrication Symposium*. Austin,Texas, 2013, pp. 260–266.
[9] Van Loock W, Pipeleers G, Diehl M, et al. Optimal path following for differentially flat robotic systems through a geometric problem formulation. *IEEE Trans Robot* 2014; 30: 980–985.
[10] Jin Y, Du J, He Y. Optimization of process planning for reducing material consumption in additive manufacturing. *J Manuf Syst* 2017; 44: 65–78.
[11] Ding D, Pan Z, Cuiuri D, et al. A tool-path generation strategy for wire and arc additive manufacturing. *Int J Adv Manuf Technol* 2014; 73: 173–183.
[12] Zaeh MF, Branner G. Investigations on residual stresses and deformations in selective laser melting. *Prod Eng* 2010; 4: 35–45.
[13] Dewil R, Vansteenwegen P, Cattrysse D. A review of cutting path algorithms for laser cutters. *Int J Adv Manuf Technol* 2016; 87: 1865–1884.
[14] Livesu M, Attene M, Spagnuolo M, et al. *A study of the state of the art of process planning for additive manufacturing*http://bibliograzia.imati.cnr.it/reports/study-state-art-process-planning-for-additive-manufacturing (2016).
[15] The MathWorks Inc. MATLAB 14b.