ACTA IMEKO
ISSN: 2221-870X
February 2015, Volume 4, Number 1, 5 - 10

An advanced GCode analyzer to predict build-time in AM components

Luca Di Angelo, Paolo Di Stefano and Emanuele Guardiani.

University of L'Aquila, Department of Industrial and Information Engineering and Economics, Monteluco di Roio, 67100 L'Aquila, Italy.

ABSTRACT
Additive Manufacturing (AM) is a technology for quickly fabricating physical models, functional prototypes and small batches of parts, by stacking two-dimensional layered features, directly from computer-aided design (CAD) data.
One of the most important challenges in this sector is represented by the capability to predict in advance the build-time because it's crucial to evaluate production cost. In this paper, an accurate method for obtaining build-time is proposed. This method is based on an advanced GCode analyzer written in Python following an Object-Oriented paradigm for scalability and maintainability. Some examples were used to demonstrate the reliability of the algorithm and possible use-cases of it are illustrated.

Section: RESEARCH PAPER

Keywords: Journal; template; IMEKO; Microsoft Word

Citation: Thomas Bruns, Dirk Röske, Paul P.L. Regtien, Francisco Alegria, Template for an IMEKO event paper, Acta IMEKO, vol. 3, no. 1, article 1, January 2014, identifier: IMEKO-ACTA-03 (2014)-01-01

Editor: Paolo Carbone, University of Perugia, Italy

Receivedmonthday, year; **In final form**month day, year; **Published**January2014

Funding: This work was supported by Measurement Science Consultancy, The Netherlands

Corresponding author: Emanuele Guardiani, e-mail: emanuele.guardiani@graduate.univaq.it

1. INTRODUCTION

AM technologies are growing day by day, thanks to the several vantages which characterize them [1]. They allow to create designed objects using new and innovative shapes, which could not be obtained by the traditional manufacturing process [2]. Is this the case of shape optimization of structural component in which the unstressed material could be removed to reduce the component weight [3], [4]. Moreover, since the AM processes are well-integrated with CAD instruments, the pre-processing for the physical creation of a part requires a shorter time compared with classical subtractive technologies. This property represents a key-feature for companies that work in a competitive business, for which the reduction of the Time to Market determines the competitivity of the company itself.
Nevertheless, AM technologies are still expensive and the manufacturing process requires a time significantly longer than the classical subtractive manufacturing technologies [5]. In a competitive market of AM services, manufacturing costs must be estimated in a reliable way [6]–[8]. That's the reason why an accurate estimation of build-time is mandatory. Reliable quantification of the build-time serves also to implement a method devoted to finding the deposition direction which minimizes manufacturing costs [9]–[23].

In this aim, several research activities were conducted during the last years. These efforts have led to two different strategies for estimating the build-time.

A first strategy performs a detailed–analysis of the manufacturing activity and should be also the most reliable.

A second approach performs the build-time estimation by properly-defined parametric functions in which the independent variables are few build-time driving factors.

The detailed-analysis based methods use complete information related to the geometry of the object and to the manufacturing process. By this method, a very accurate estimation of the build-time could be potentially obtained. On the other side, the complete information is necessary for that evaluation and a detailed analysis of the manufacturing activity is required. For these reasons, detailed-analysis is computationally expensive and more time is required for build-time estimation.

Instead the parametric methods are less accurate than the first one but, on the other hand, they require few data as input and they are computationally less expensive. The parametric-based methods make use of some driving build-time factors, which are the independent parameters used in the functions to evaluate the build-time, which is the dependent variable. These parameters can be computed by analysing the geometric model of the object to be manufactured (volume, bounding box, etc.). The most challenging aspect of this typology of methods consists in identifying these factors. The set of parameters used should take into account all the elements which affect build-time and they must be independent each other so that any cross-correlation is avoided. The parametric methods appear more promising for many practical applications such as in those contexts where an accurate prediction of the build-time is required but a limited set of data describing the object is available. It is the case of the budgeting process, where the customer may not have the intention of providing the full geometrical model of the object to the seller, in order to protect its intellectual property, but can provide just some parameters that affect cost [24]. Moreover, the build-time estimation is a mandatory step for any optimization method devoted to search the best build direction. It is for these reasons that parametric approaches have been proposed generally as a component of much more articulated methods devoted to search the optimal manufacturing build direction. [25] suggests a very simplified model in which the build-time is proportional to the number of layers in the sliced model. [21], [23], [24], [26]–[28] propose more complex formulations of the build-time, which is dependent on the volume of the object, of the supports and other geometrical features of the object. Although they use many more parameters, the limitation of these methods is in the function describing the build-time, which is linear. The relationship between build-time and its driving factors is very complex and unknown. In order to take into account this problem, [29] suggests using an adaptive model based on the Grey Modelling, while [30] proposes an Artificial Neural Network (ANN) for identifying the relationship between the driving factors and the build-time. The authors demonstrate that in this way a very accurate estimation of the build-time can be obtained. The limit of both these approaches is they need a large set of model samples. In order to provide a set of training samples, a lot of driving build-time factors and related real build-time should be examined. The accuracy of estimation performed by the neural network, for example, increases with the number of test samples, so that a large number of real build-time evaluations is required. Obviously it would not be economically and temporally feasible.

In order to define a significative set of samples, an accurate detailed-analysis based method could be used, which performs a less expansive evaluation of the build-time with respect to a real experiment. A detailed-analysis can be operated by using specialized CAE programs such as those supplied with the AM machine. It is the case of Simplify 3D, which is a 3D printing slicing software that controls every aspect of the printing process and performs also build-time evaluation. Some alternatives to this tool are Cura by Ultimaker and MatterControl by MatterHackers, both freeware. Nevertheless the build-time estimations provided by these software is different from those performed with the FDM machine (German RepRap X350). In Table 1 the comparison of the estimated build-time with the real one is reported, using three test cases shown in Figure 1.

Table 1. Comparison between real build-time and an estimation provided by professional CAE software for the models reported in Figure 1.

Model	Software	Real	Estimated	*Error*
A	Simplify3D	1532 min	1344 min	*-14 %*
B	Cura	374 min	242 min	*-54 %*
C	MatterControl	326 min	286 min	*-14 %*

The estimation performed is not accurate enough to be used for obtaining qualified training data for a neural network. The main reason which led to the mismatch evidenced in the previously mentioned programs do not consider some process-related parameters of the machine in use, which are acceleration and jerk.

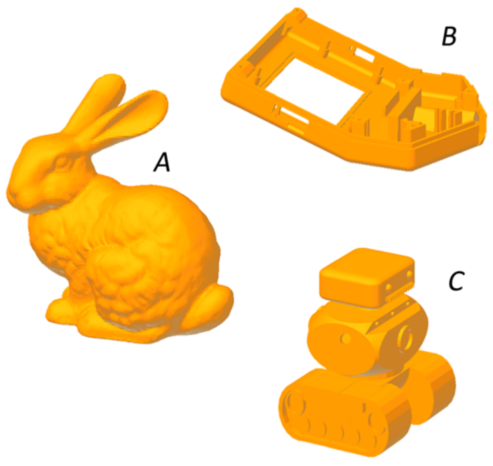

Figure 1. Reference models used for time comparison.

The control strategy performed by some machine firmware, such as the open-source RepRap, uses acceleration strategies, defined by two parameters "jerk" and "acceleration" that widely affect build-time and object quality. This is shown in Figure 2, where two cubes of the same dimension are manufactured, by a German RepRap X350, using different acceleration parameters. In Figure 2(a) a module of 300 mm/s^2 was selected for the x-y axis, while in Figure 2(b) twice of the value than the previous one is used.

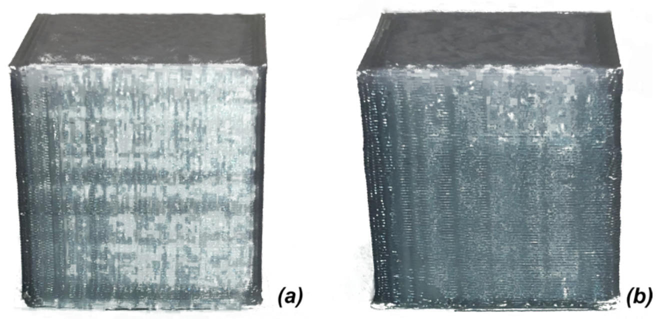

Figure 2. Comparison between two cubes, based on the same nominal geometry and manufactured using a different acceleration value.

A comparison between the two objects shows that the quality of the corners of the model (a) is significantly better than the model (b). This is due to the different inertia forces, which affect the accuracy of the material deposition performances, especially on the corners.

In order to overcome all the previously discussed limitations of the implemented detailed-analysis based methods, in this paper, a new method for build-time evaluation has been developed. The followed approach has been quite a general-purpose one, so that this method can be used in several AM applications. The method performs the GCode analysis, taking into account the most important process-related parameters. In this way, a very accurate estimation of the real build-time can be performed. In the next sections, the method will be explained in detail. Some test cases will be used to test the proposed approach and the results will be critically discussed.

1.1. GCode

The GCode is the most widespread programming language used for giving instructions to Computer Numerical Control (CNC) machine. It consists of a series of textual instructions (word address), in accordance with the standard ISO 6983, that regulate the behavior of the machine during the manufacturing process. For example, defining the speed of the axis, regulating the temperature of the extruder and much more. This set of commands is generally called Part Program. During the "first" era of the automatic machines, a manual approach was adopted for generating the Part Program. An high-skilled operator was able to write the Part Program by itself or, at the best, with the aid of visual programming software. Today, instead, the preferred solution consists of an automatic approach based on the integration between the CAD and CAM systems (Figure 3). After the design of the object, the geometry is imported into the CAM, where many of the process-related parameters are defined. With reference to AM technologies, these parameters could be represented by the layer thickness, the typology of hatching, the speeds of the motors and much more. Then, if required by the technology in use, the supports are generated too into the CAM, typically using a graphical procedure. At the end of these steps, the CAM generates the tool path and the Post Processor transforms all the previous information in a set of instructions, readable by the machine in use, which is the Part Program, saved into a textual file that is finally sent to the machine.

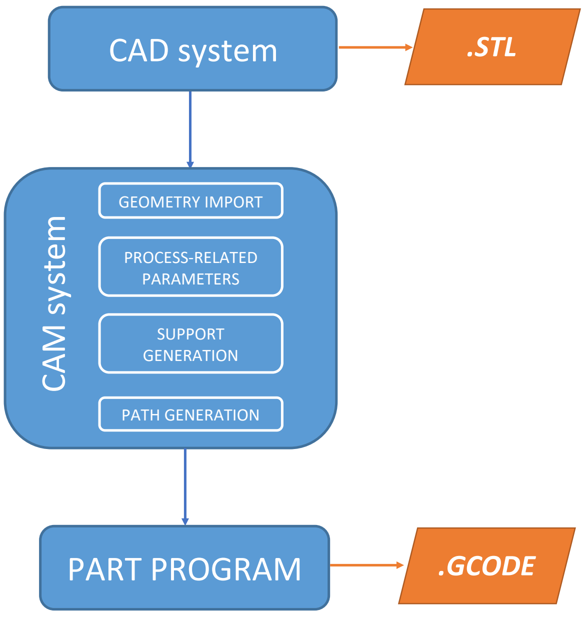

Figure 3. Automatic programming for generating the Part Program.

Consequently, the Part Program defines many characteristics of the manufacturing process that will be followed by the machine. But an analysis limited to the Part Program is generally not sufficient to establish the kinematic behavior of the machine. Many decisions about that are taken in real-time by the controller of the machine. This is the case for acceleration ramps, which are tightly dependent on the machine in use. Any CNC machine, due to the axis inertia and/or torque availability of the motors, is subject to magnitude constraints on accelerations. Three different control strategies may be adopted. Supposing that the deposition tool has to follow the path reported in blue in Figure 4, where *p1*, *p2,* and *p3* are three control points interpolated linearly. A first option consists of maintaining a fixed speed along the entire path. This condition represents the optimal solution in terms of time-saving but, on the other side, a similar behavior can be realized only theoretically due to the infinite acceleration module required to realize that. Actually, when a fixed speed is adopted (Figure 5), a diverted path (green line of Figure 4) is generated due to the impossibility of realizing an infinite module acceleration. In order to avoid a similar behavior that causes aleatory geometrical errors on the finite piece, another strategy, which is characterized by having a zero speed in correspondence of control points (blue-dotted line in Figure 5), can be used. In this case, linear acceleration ramps are generally adopted. This allows obtaining the optimal positioning accuracy, being the theoretic path and the real path the same.

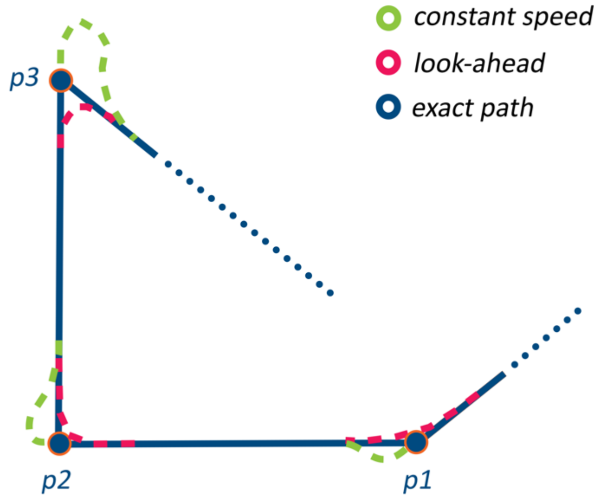

Figure 4. Comparison of tool paths using different acceleration strategies.

The drawback of this strategy is given by the increase of the build-time. This trouble has led to formulate a third approach, which represents a compromise solution between geometrical accuracy and build-time. In this case, the control unit of the machine in use does not limit itself to execute the instructions provided by the GCode but it performs a "clever" evaluation. In addition to the command in execution, some next instructions are read and cached. Then, through a vectoral comparison of the velocities, the acceleration ramps are computed in order to maintain the maximum feedrate possible (red-dotted line Figure 5). This control strategy is typically known as "look-ahead" and it can be regarded as a multi-objective optimization problem, where the geometrical accuracy conflicts with the reduction of the build-time.
Many researchers have worked on this topic during the last years. Since the related literature is wide and voluminous, in the following some of the most important and recent methods are analyzed. [31] proposes that each corner is smoothed by replacing a subset of the path that contains it with a conic "splice" segment, deviating from the exact corner by no more than a prescribed tolerance. [32] suggests using a fine-interpolation parametric method in which the corners are replaced by arc curves, while in [33], [34] the B-splines are used to approximate the corners.

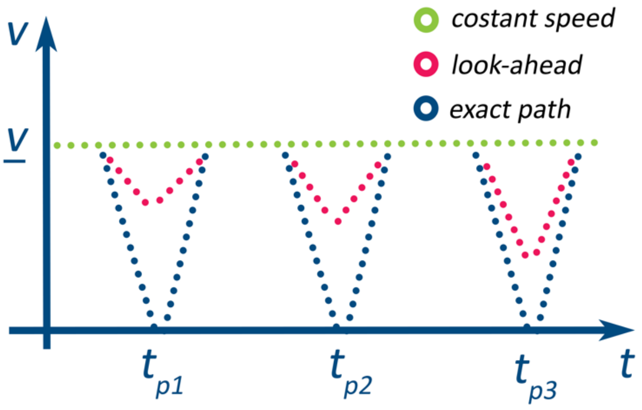

Figure 5. Speed comparison between constant speed, zero-velocity, and look-ahead strategies.

In [35] a different kind of approach is used: the control acts on the acceleration ramps instead of the tool paths. The presence of so many different approaches to the "look-ahead" optimization can generate some confusion. That's the case of FDM machines for which, as we verified in the previous section for the case of a German RepRap X350, also professional tools are not able to reconstruct the real kinematic behavior of the machine, leading to a very rough estimation of the build-time.

In this paper, a deep investigation of the behavior of RepRap devices is computed. The kinematic laws are reconstructed and, lastly, a script for estimating very accurately the build-time, written in Python following an Object-Oriented Paradigm for scalability and maintainability, is provided.

2. PROPOSED METHOD

The proposed method is based on the analysis of the GCode, whose role was previously described. In particular, all the instructions which are referred to movement command are considered, being these the main responsible for defining the build-time. Other instructions that may contribute to the build-time are the temperature commands that, for example, define the temperature of the printing bed. But for this category of instructions, an exact evaluation of how they contribute to the build-time is hard to provide because they are closely depended on environmental conditions. Moreover, their influence is generally negligible with respect to movement commands and so they will not be considered.

The GCode movement commands are described in *Annex E* of ISO 6983, therefore they will not be discussed in detail here. As already illustrated in Figure 3, being the GCode, used for AM applications, generated starting from a triangular mesh (.stl) that is afterward sliced, most of the analytical information which defines the original geometry is lost during the data exchange. Consequently, the simplest and most efficient way for generating the Part Program is to define many geometrical control points that belong to the model and interpolate them linearly. Therefore, most of the movement instructions used for AM applications are linear interpolations, introduced by the code *G01*. The algorithm evaluates each interpolation line using a regular expression matching. Each movement is associated with its relative velocity and length. In Figure 6, as a sample, the interpolation lines *i* and *i+1* of a generic GCode file are reported.

```
i)      G1 Xaaa.aaa Ybbb.bbb Ec.cc Fdddd
i+1)    G1 Xeee.eee Yfff.fff Eg.gg
```

Figure 6. Sample of two linear interpolation's GCode instructions.

The length l_i of path *i-th*, the nominal speed $v1_i$ that should be reached and the length of extruded filament e_i during the *i-th* step can be formulated as follows.

$$l_i = \sqrt{(eee.eee - aaa.aaa)^2 + (fff.fff - bbb.bbb)^2}$$

$$v1_i = dddd$$

$$e_i = g.gg$$

For each step *i-th* the build-time is provided by the contribution of four terms:

$$t_i = t_{JERKi} + t_{0i} + t_{1i} + t_{2i} \qquad (1)$$

Where:

t_{JERKi}: jerk phase time during *i-th* step
t_{0i}: acceleration phase time during *i-th* step
t_{1i}: cruise phase time during *i-th* step
t_{2i}: deceleration phase time during *i-th* step

To evaluate these terms, the acceleration phase and the jerk phase need to be here discussed and illustrated.

The jerk phase, in particular, is referred to the "look-ahead" strategy in use by RepRap machines. During that phase, an "instantaneous" change of speed Δv is applied. This particular behavior is made possible by using of stepper motors. Obviously, "instantaneous" change of velocity implies high values of acceleration, so the value of Δv must be limited to avoid both vibrations and a loss of steps from the motors. Being the contribution of t_{JERKi} generally very small compared to the other three terms of formula (1), it is assumed to be negligible.

$$t_{JERKi} \cong 0 \qquad (2)$$

In order to proceed, for each step, a generic *j-th* degree of freedom, which satisfies the condition (3), is taken as reference.

$$a_i^j \neq 0 \qquad j = 1, 2, \dots, N \qquad (3)$$

a_i^j: acceleration module of *i-th* step referred to the *j-th* degree of freedom

where N identifies the number of degrees of freedom of the machine. That assumption is justified by the synchronous behavior which should be carried out by the axis: the axis should start and stop to execute the command of a certain instruction line at the same moment. The following condition is then evaluated.

$$\left|v2_i^j - v0_{i+1}^j\right| > \Delta v_i^j \qquad (4)$$

$v2_i^j$: end speed of step *i-th* refereed to the *j-th* degree of freedom

$v0^j_{i+1}$: start speed of step *(i+1)-th* refereed to the *j-th* degree of freedom

$v2^j_i$ and $v0^j_{i+1}$ are initially computed taking as reference the feedrate value reported into the GCode instructions and they are generally different each other. This is due to the fact that from step *i-th* to step *(i+1)-th* the vector $\boldsymbol{v1}$ could change in direction and/or magnitude. If the condition (4) it's true, end speed $v2^j_i$ or/and start speed $v0^j_{i+1}$ need to be revaluated. In this case, two possibilities are defined. If

$$v2^j_i \cdot v0^j_{i+1} > 0 \tag{5}$$

then just one of the two velocities requires to be updated.

$$\begin{cases} \boldsymbol{v2^*_i} = \dfrac{v0^j_{i+1} + \Delta v^j_i}{v2^j_i}\boldsymbol{v2_i} & |v2^j_i| > |v0^j_{i+1}| \\ \boldsymbol{v0^*_{i+1}} = \dfrac{v2^j_i + \Delta v^j_i}{v0^j_{i+1}}\boldsymbol{v0_{i+1}} & |v2^j_i| < |v0^j_{i+1}| \end{cases} \tag{6}$$

Otherwise, both velocities are recalculated in the following way.

$$\begin{cases} \boldsymbol{v2^*_i} = \dfrac{\Delta v^j_i}{v2^j_i + v0^j_{i+1}}\boldsymbol{v2_i} \\ \boldsymbol{v0^*_{i+1}} = \dfrac{\Delta v^j_i}{v2^j_i + v0^j_{i+1}}\boldsymbol{v0_{i+1}} \end{cases} \tag{7}$$

The physical significance of the jerk phase can be better understood considering Figure 7. By assuming that the deposition tool is moving from point *A* to point *B,* then to *C'* or *C''*, maintaining a constant feedrate. In the case of point *C',* the variation in direction of the velocity vector is small. Therefore, instead of reducing the speed in point *B* to zero, just a limited deceleration is applied until a certain speed value, whose entity depends on the chosen jerk module (equation (6)). Otherwise, when the deposition tool has to follow the path *ABC''* the vectorial variation of the velocity is significant, consequently both the end speed of path *i-th* $\boldsymbol{v2_i}$ then the start speed of the path *(i+1)-th* $\boldsymbol{v0_{i+1}}$ are down-scaled (equation (7)). Anyway, it is interesting to notice that in both cases the speed in point *B* is never equal to zero. This affects the manufacturing process in terms of vibrations, quality of deposition and accuracy that, by limiting the jerk parameter, can be contained into acceptable values. On the other side, this compromise allows to obtain a drastic reduction of the build-time.

After that the jerk phase has defined the start speed $\boldsymbol{v0_i}$ and the end speed $\boldsymbol{v2_i}$ of the *i-th* step, the acceleration ramps can be computed. This kind of machine generally makes use of linear acceleration ramps, which are the most simple to handle when a digital electronic is utilized. Assuming to be at step *i-th*, four different kinds of situations may occur from GCode's reading. The first one, reported in Figure 8, represents the condition for which nominal speed $v1^j_i$ is reached along the *j-th* degree of freedom. The following quantities are defined.

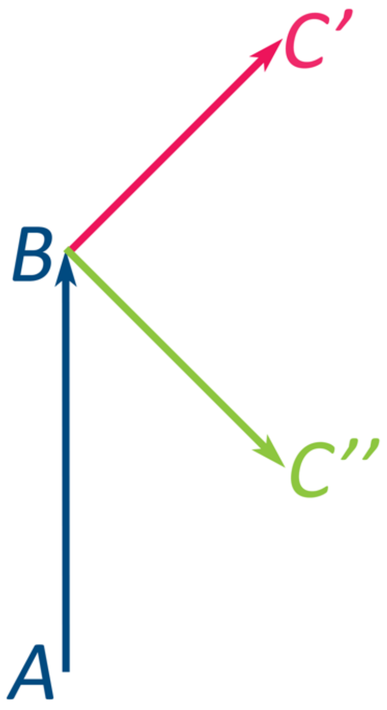

Figure 7. During the jerk phase, the start speed and end speed of the paths are "overwritten" cleverly.

$$\begin{cases} t_{0i} = \dfrac{|v1^j_i| - |v0^j_i|}{a^j_i} \\ s^j_{0i} = |v0^j_i|\, t_{0i} + \dfrac{1}{2}a^j_i\, t^2_{0i} \end{cases} \tag{8}$$

$$\begin{cases} t_{2i} = -\dfrac{|v2^j_i| - |v0^j_i|}{a^j_i} \\ s^j_{2i} = |v1^j_i|\, t_{2i} - \dfrac{1}{2}a^j_i\, t^2_{2i} \end{cases} \tag{9}$$

If the next condition (10) becomes true, the behavior illustrated in Figure 8 is performed during *i-th* step.

$$s^j_{0i} + s^j_{2i} > s^j_i \tag{10}$$

In that case t_{1i} and s^j_1 can be computed as

$$\begin{cases} t_{1i} = \dfrac{s^j_1}{v1^j_i} \\ s^j_1 = s^j_i - s^j_{0i} - s^j_{2i} \end{cases} \tag{11}$$

s^j_i: total distance during *i-th* step along the *j-th* degree of freedom

s^j_{0i}: acceleration distance during *i-th* step along the *j-th* degree of freedom

s^j_{1i}: cruise distance during *i-th* step along the *j-th* degree of freedom

s^j_{0i}: deceleration distance during *i-th* step along the *j-th* degree of freedom

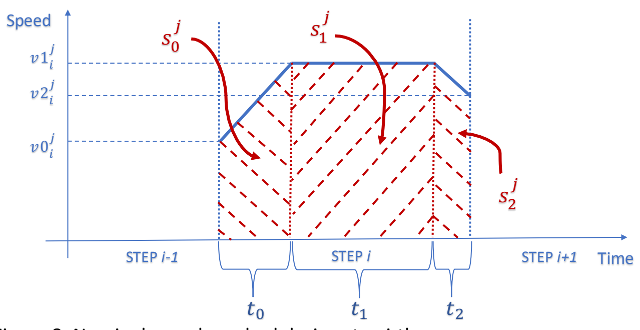

Figure 8. Nominal speed reached during step i-th.

Other cases can happen when the nominal speed $v1_i^j$ along the *j-th* degree of freedom can't be reached due to an insufficient path length s_i^j. In such a case we can distinguish three different possibilities. The first one is illustrated in and occurs when the conditions (12) become true.

$$\begin{cases} s_{0i}^j > s_i^j \\ v0_i^j < v2_i^j \end{cases} \tag{12}$$

$$\begin{cases} t_{0i} = \dfrac{|v2_i^j| - |v0_i^j|}{a_i^j} \\ s_{0i}^j = |v0_{j_i}| t_{0i} + \dfrac{1}{2}\, a_i^j t_{0i}^2 \end{cases} \tag{13}$$

In this case a new end speed $v2_i^j$ is necessary and, consequently, condition (4) must be re-evaluated between step *i-th* and step *(i+1)-th*. This particular behavior requires to use a "closed-loop" control. For each *i-th* step, updated evaluations are performed for the *(i-1)-th* and *(i+1)-th* steps too, if required. For this reason, the step elaborated by the "look-ahead" algorithm is a little ahead respect to the step which the machine is being realizing. New end speed $v2_i^{j*}$ and acceleration time t_{0i}^* are evaluated in the following way.

$$\begin{cases} v2_i^{j*} = \sqrt{2\, a_i^j s_i^j + |v0_i^j|} \\ \boldsymbol{v2_i^*} = \dfrac{v2_i^{j*}}{v1_i^j} \boldsymbol{v2_i} \\ t_{0i}^* = \dfrac{|v2_i^{j*}| - |v0_i^j|}{a_i^j} \end{cases} \tag{14}$$

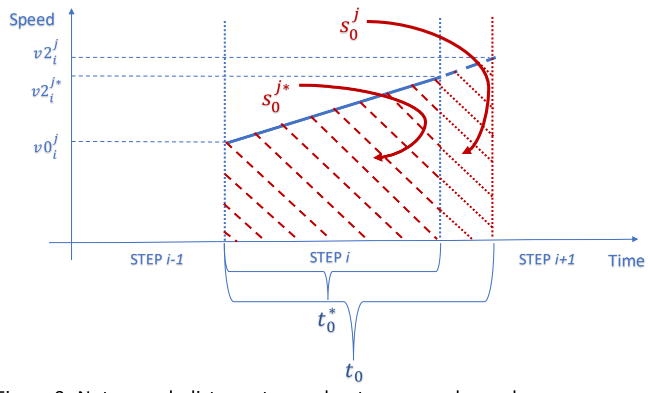

Figure 9. Not enough distance to accelerate, new end speed.

An opposite situation is illustrated in Figure 10. In this case, there is not enough space for decelerating and, consequently, the start speed $v0_i^j$ needs to be updated. That occurs when the conditions (15) are verified.

$$\begin{cases} s_{2i}^j > s_i^j \\ v0_i^j > v2_i^j \end{cases} \tag{15}$$

$$\begin{cases} t_{2i} = \dfrac{|v2_i^j| - |v0_i^j|}{-a_i^j} \\ s_2^j = |v0_{j_i}| t_{2i} - \dfrac{1}{2}\, a_i^j t_{2i}^2 \end{cases} \tag{16}$$

So, similarly to the previous one, new start speed $v0_i^{j*}$ and deceleration time t_{2i}^* can be formulated as follows.

$$\begin{cases} v0_i^{j*} = \sqrt{2\, a_i^j s_i^j + |v2_i^j|} \\ \boldsymbol{v0_i^*} = \dfrac{v0_i^{j*}}{v1_i^j} \boldsymbol{v0_i} \\ t_{2i}^* = \dfrac{|v2_i^j| - |v0_i^{j*}|}{a_i^j} \end{cases} \tag{17}$$

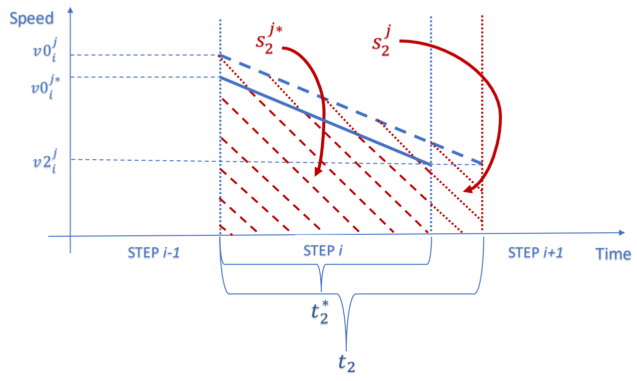

Figure 10. Not enough space to decelerate, new start speed.

A last possible situation is identified in Figure 11. In this case, nominal speed $v1_i^j$ is not reached but there is enough distance for accelerating and decelerating. That behavior is realized when the condition (18) is verified.

$$s_{0i}^j + s_{2i}^j > s_i^j \tag{18}$$

$$\begin{cases} t_{0i} = \dfrac{|v1_i^j| - |v0_i^j|}{a_i^j} \\ s_{0i}^j = |v0_i^j|\, t_{0i} + \dfrac{1}{2} a_i^j\, t_{0i}^2 \end{cases} \tag{19}$$

$$\begin{cases} t_{2i} = -\dfrac{|v2_i^j| - |v0_i^j|}{a_i^j} \\ s_{2i}^j = |v1_i^j|\, t_{2i} - \dfrac{1}{2} a_i^j\, t_{2i}^2 \end{cases} \tag{20}$$

For this case a new cruise speed $v1_i^{j*}$ is defined.

$$v1_i^{j*} = \sqrt{\frac{{v0_i^j}^2 + {v2_i^j}^2}{2\,a_i^j} + a_i^j s_i^j} \tag{21}$$

After calculating $v1_i^{j*}$ as suggested by (21), the terms t_{0i} and t_{2i} can be finally evaluated.

$$\begin{cases} t_{0i} = \dfrac{|v1_i^{j*}| - |v0_i^j|}{a_i^j} \\ s_{0i}^j = |v0_i^j|\, t_{0i} + \dfrac{1}{2} a_i^j\, t_{0i}^2 \end{cases} \tag{22}$$

$$\begin{cases} t_{2i} = -\dfrac{|v2_i^j| - |v1_i^{j*}|}{a_i^j} \\ s_{2i}^j = |v2_i^j|\, t_{2i} + \dfrac{1}{2} a_i^j\, t_{2i}^2 \end{cases} \tag{23}$$

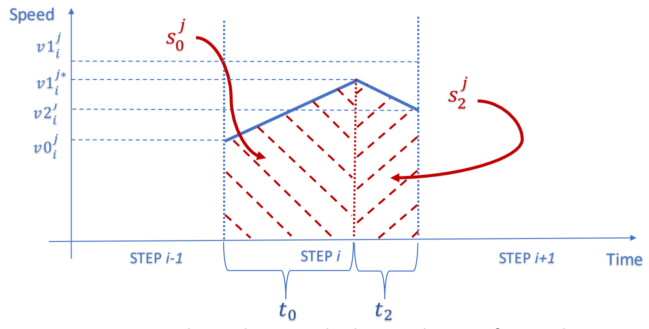

Figure 11. Nominal speed not reached. Enough space for accelerating and decelerating.

The previous relations and considerations were implemented into a python script, whose flowchart is described in Figure 12. At the beginning the software evaluates the total number of instructions contained into the Part Program. Then, using a regular expression matching, the movement commands are identified and for each of them, an associate speed and length is pulled out and memorized. The next phases consist of applying, for each movement command, the Jerk and Acceleration phases, accordingly with the relationships previously illustrated. At the end of the cycle the total build-time, estimated by the algorithm, is provided.

3. RESULTS

In order to validate and refine the model, some comparisons were made between theoretical build-time and real-time required by a FDM machine (German RepRap X350). Nine objects, represented in Figure 13, which are characterized by different topological and geometrical features, were chosen for the experiment. Every part was physical realized to obtain the real value of the build-time. Different CAE software was used for generating the Part Program and their estimation were compared both with the real build-time and its estimation calculated by the proposed algorithm. Because each CAE software could implement a different slicing strategy, the Part Program of the same part obtained using different automatic procedures can not be compared, also if the same process parameters are used. Therefore for each test case only one of the three CAE software considered (Cura, MatterControl and Simplify3D) were used. In Table 2 the parameters used for the experiment are reported.

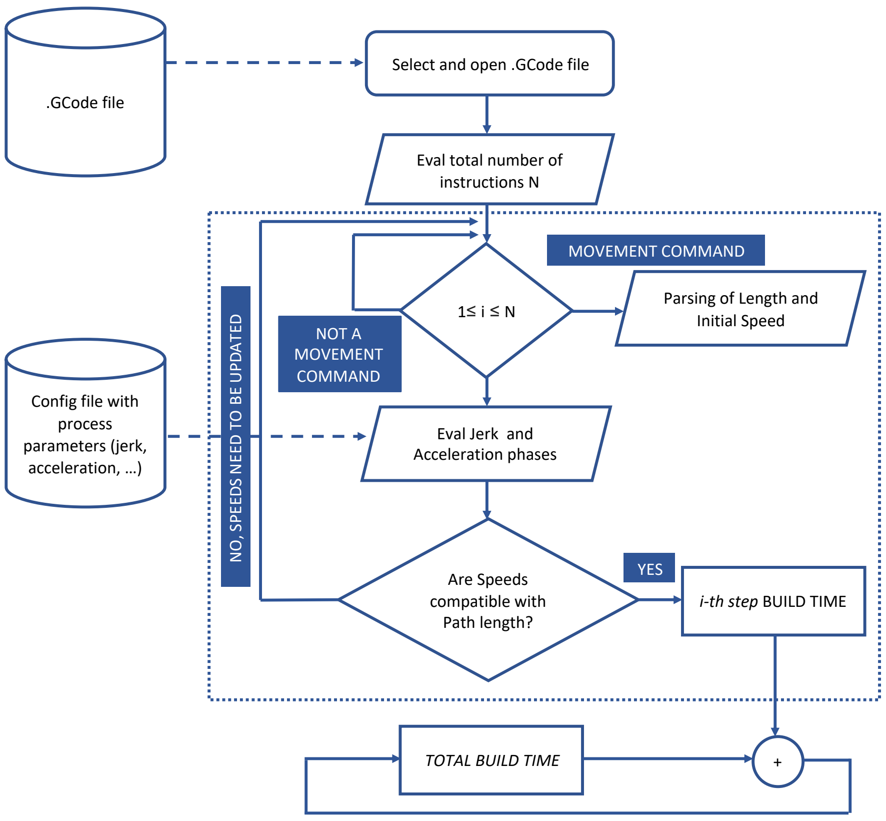
.GCode file
Select and open .GCode file
Eval total number of instructions N
MOVEMENT COMMAND
1≤ i ≤ N
Parsing of Length and Initial Speed
NOT A MOVEMENT COMMAND
NO, SPEEDS NEED TO BE UPDATED
Config file with process parameters (jerk, acceleration, ...)
Eval Jerk and Acceleration phases
Are Speeds compatible with Path length?
YES
i-th step BUILD TIME
TOTAL BUILD TIME
+

Figure 12. Algorithm flow-chart.

Table 2. Process related parameters used for the experiment.

Setting	Value		
Acceleration	300 mm/min² (slow)	1000 mm/min² (rapid)	
Jerk (Δv)	18 mm/min		
Layer thickness	0,1 mm		
Maximum speed	100 mm/s (X-Y)	10 mm/s (Z)	100 mm/s (E)
Retract length	2 mm		

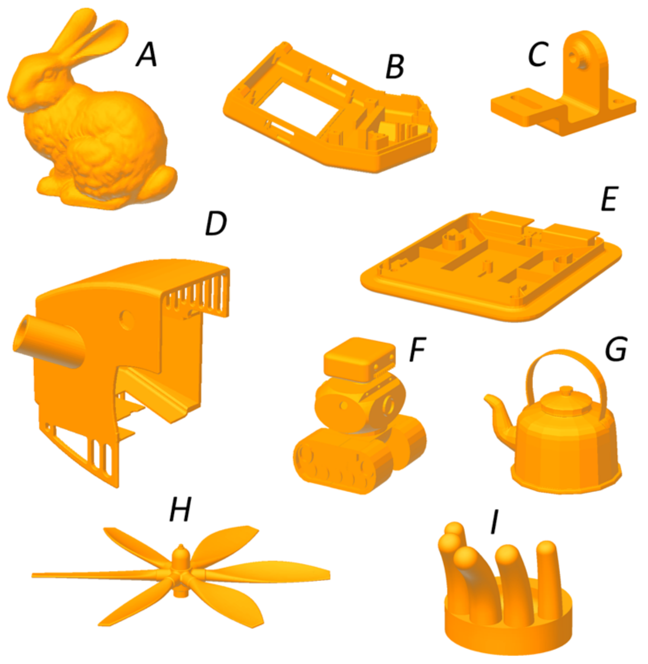

Figure 13. Models used for analyzing the accuracy of the proposed algorithm.

The results of the tests are illustrated in Table 3. It can be noticed that, for each of the used CAE software, the error values cannot be neglected. In particular, for the platform Cura the average error for the three analyzed test cases amounts to 54 % of the real build-time. A better situation is found by using Simplify 3D, which shows an average error of 15 % with respect to the real build-time. Anyway, a similar error can not be accepted in most of the applications. Finally, it can be affirmed that the optimal behavior is represented by MatterControl. The average error for that software, in fact, is 8 % of the real build-time. A similar value could be accepted in some contests such as, for example, the budgeting process. Anyway, two aspects should be noticed. As first, the error for the test case *F* is equal to 14 % of the real build-time. This shows that the error is not characterized by a well-known trend, so the level of confidence of the estimations is low. Moreover, a parametric method such as [36] is able to provide a better estimation of the real build-time requiring, at the same time, a significantly less amount of time for the set-up.

On the other side, the valuation obtained using the proposed algorithm is very satisfactory, the error is negligible (≤ 1 %) in any case of study. For parts *B, E* and *F* the estimation is even equal to the real build-time. These results underline that the proposed method evaluates the real behavior of RepRap machines. Moreover all the causes which systematically affect the build-time are identified and taken into account.

4. CONCLUSIONS

In this paper, a new method, which performs accurate estimation of the build-time, has been proposed. This result is possible thanks to an advanced analysis of the Part Program of the object to be manufactured integrated with information concerning control strategies.

The kinematic behavior of a CNC machine is defined partially by the Part Program, since the tool movements are determined also by the control strategies implemented in the machine. It is the case of so-called “look-ahead” algorithm, implemented to control the machine movements, aimed to reduce the build-time, maintaining at the same time the geometrical and dimensional quality of the final object.

It has been noticed that also professional CAE tools, developed for AM applications, are not able to evaluate accurately the build-time required for fabricating objects realized using additive technologies. This results look so negative because they are obtained from the software that performs the manufacturing process from the given geometric data (.stl file)

Therefore, with the aim of testing the procedure for determining the build-time, the case study of RepRap was taken as reference, since it is a very diffused controller for AM machines. By a detailed-analysis of the RepRap machines, the control strategies were identified and reproduced inside the proposed method. In particular, it provides a mathematical formulation of the “look-ahead” control strategy (“jerk phase”).

In order to compute the build-time, a custom python application was developed. As shown in the results section, the method provides a very accurate estimation of the build-time. For each of the nine test cases analyzed, the error is less than 0,2 % of the real build-time, but in some cases the estimated and real build-time are the same.

Table 3. Results of the experiment.

Part	Volume [cm^3]	Real build-time [min]	Software	CAE build-time [min]	Error	Proposed method [min]	Error
A	210	1532	Simplify 3D	1344	-14 %	1535	0,2 %
B	23	374	Cura	242	-54 %	373	-0,1 %
C	14	156	Simplify 3D	146	-7 %	156	0 %
D	127	1206	Simplify 3D	976	-24 %	1205	-0,01 %
E	27	402	MatterControl	388	-4 %	402	0 %
F	21	326	MatterControl	286	-14 %	326	0 %
G	245	2177	MatterControl	2040	-7 %	2173	-0,2 %
H	7	101	Cura	64	-58 %	101	0 %
I	25	257	Cura	170	-51 %	257	0 %

A limit of this method is it needs the Part Program which is obtained at the end of several steps (Figure 3). Therefore, this approach is quite time-consuming and could not represent the optimal solution when the evaluation should be provided quickly and some information is still missing. For example, some specific parameters of the machinery in use, such as acceleration and jerk. In these cases, the optimal solution is still represented by the parametric methods, for which the proposed method can represent a powerful instrument for the set-up.

Further work is required to test the proposed methodology for other typologies of AM machines that does not implement RepRap controller and use other control strategies.

REFERENCES

[1] K. V. Wong and A. Hernandez, "A Review of Additive Manufacturing," ISRN Mech. Eng., 2012.

[2] S. S. Babu and R. Goodridge, "Additive manufacturing," *Materials Science and Technology (United Kingdom)*, vol. 31, no. 8. Maney Publishing, pp. 881–883, 01-Jun-2015.

[3] D. Brackett, I. Ashcroft, and R. Hague, "Topology optimization for additive manufacturing," in *22nd Annual International Solid Freeform Fabrication Symposium - An Additive Manufacturing Conference, SFF 2011*, 2011.

[4] F. Cucinotta, M. Raffaele, and F. Salmeri, "A stress-based topology optimization method by a Voronoi tessellation Additive Manufacturing oriented," Int. J. Adv. Manuf. Technol., vol. 103, Aug. 2019.

[5] D. S. Thomas and S. W. Gilbert, "Costs and cost effectiveness of additive manufacturing: A literature review and discussion," in Additive Manufacturing: Costs, Cost Effectiveness and Industry Economics, 2015.

[6] P. Alexander, S. Allen, and D. Dutta, "Part orientation and build cost determination in layered manufacturing," CAD Comput. Aided Des., 1998.

[7] G. Costabile, M. Fera, F. Fruggiero, A. Lambiase, and D. Pham, "Cost models of additive manufacturing: A literature review," Int. J. Ind. Eng. Comput., 2016.

[8] S. L. Chan, Y. Lu, and Y. Wang, "Data-driven cost estimation for additive manufacturing in cybermanufacturing," J. Manuf. Syst., 2018.

[9] E. Asadollahiyazdi, J. Gardan, and P. Lafon, "Multi-Objective Optimization of Additive Manufacturing Process," IFAC-PapersOnLine, vol. 51, pp. 152–157, Jan. 2018.

[10] K. Thrimurthulu, P. M. Pandey, and N. V. Reddy, "Optimum part deposition orientation in fused deposition modeling," Int. J. Mach. Tools Manuf., 2004.

[11] Y. Zhang, A. Bernard, R. K. Gupta, and R. Harik, "Feature based building orientation optimization for additive manufacturing," Rapid Prototyp. J., 2016.

[12] Y. Zhang, A. Bernard, R. Harik, and K. P. Karunakaran, "Build orientation optimization for multi-part production in additive manufacturing," J. Intell. Manuf., 2017.

[13] P. Das, R. Chandran, R. Samant, and S. Anand, "Optimum Part Build Orientation in Additive Manufacturing for Minimizing Part Errors and Support Structures," in *Procedia Manufacturing*, 2015.

[14] S. Chowdhury, K. Mhapsekar, and S. Anand, "Part Build Orientation Optimization and Neural Network-Based Geometry Compensation for Additive Manufacturing Process," J. Manuf. Sci. Eng. Trans. ASME, 2018.

[15] G. Strano, L. Hao, R. M. Everson, and K. E. Evans, "Multi-objective optimization of selective laser sintering processes for surface quality and energy saving," in *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 2011.

[16] V. Canellidis, J. Giannatsis, and V. Dedoussis, "Genetic-algorithm-based multi-objective optimization of the build orientation in stereolithography," Int. J. Adv. Manuf. Technol., 2009.

[17] S. K. Singhal, P. K. Jain, P. M. Pandey, and A. K. Nagpal, "Optimum part deposition orientation for multiple objectives in SL and SLS prototyping," Int. J. Prod. Res., 2009.

[18] A. Li, Z. Zhang, D. Wang, and J. Yang, "Optimization method to fabrication orientation of parts in fused deposition modeling rapid prototyping," in *2010 International Conference on Mechanic Automation and Control Engineering, MACE2010*, 2010.

[19] P. Jaiswal, J. Patel, and R. Rai, "Build orientation optimization for additive manufacturing of functionally graded material objects," Int. J. Adv. Manuf. Technol., 2018.

[20] N. Padhye and K. Deb, "Multi-objective optimisation and multi-criteria decision making in SLS using evolutionary approaches," Rapid Prototyp. J., 2011.

[21] S. Khodaygan and A. H. Golmohammadi, "Multi-criteria optimization of the part build orientation (PBO) through a combined meta-modeling/NSGAII/TOPSIS method for additive manufacturing processes," Int. J. Interact. Des. Manuf., 2018.

[22] A. M. Phatak and S. S. Pande, "Optimum part orientation in rapid prototyping using genetic algorithm," in *Transactions of the North American Manufacturing Research Institution of SME*, 2012.

[23] S. E. Brika, Y. F. Zhao, M. Brochu, and J. Mezzetta, "Multi-Objective Build Orientation Optimization for Powder Bed Fusion by Laser," J. Manuf. Sci. Eng. Trans. ASME, 2017.

[24] L. Di Angelo and P. Di Stefano, "Parametric cost analysis for web-based e-commerce of layer manufactured objects," Int. J. Prod. Res., 2010.

[25] P. T. Lan, S. Y. Chou, L. L. Chen, and D. Gemmill, "Determining fabrication orientations for rapid prototyping with stereolithography apparatus," CAD Comput. Aided Des., 1997.

[26] H. S. Byun and K. H. Lee, "Determination of the optimal build direction for different rapid prototyping processes using multi-criterion decision making," Robot. Comput. Integr. Manuf., 2006.

[27] D. T. Pham, S. S. Dimov, and R. S. Gault, "Part orientation in stereolithography," Int. J. Adv. Manuf. Technol., 1999.

[28] I. Campbell, J. Combrinck, D. De Beer, and L. Barnard, "Stereolithography build time estimation based on volumetric calculations," Rapid Prototyp. J., 2008.

[29] Z. Yicha, A. Bernard, J. Munguia, and K. K.P., "Fast

adaptive modeling method for build time estimation in Additive Manufacturing," CIRP J. Manuf. Sci. Technol., Jun. 2015.
[30] L. Di Angelo and P. Di Stefano, "A neural network-based build time estimator for layer manufactured objects," Int. J. Adv. Manuf. Technol., 2011.
[31] C. A. Ernesto and R. T. Farouki, "High-speed cornering by CNC machines under prescribed bounds on axis accelerations and toolpath contour error," Int. J. Adv. Manuf. Technol., 2012.
[32] Y. A. Jin, Y. He, J. Z. Fu, Z. W. Lin, and W. F. Gan, "A fine-interpolation-based parametric interpolation method with a novel real-time look-ahead algorithm," CAD Comput. Aided Des., 2014.
[33] S. Sun, H. Lin, L. Zheng, J. Yu, and Y. Hu, "A real-time and look-ahead interpolation methodology with dynamic B-spline transition scheme for CNC machining of short line segments," Int. J. Adv. Manuf. Technol., 2016.
[34] Y. Zhang, M. Zhao, P. Ye, J. Jiang, and H. Zhang, "Optimal curvature-smooth transition and efficient feedrate optimization method with axis kinematic limitations for linear toolpath," Int. J. Adv. Manuf. Technol., 2018.
[35] L. Wang and J. Cao, "A look-ahead and adaptive speed control algorithm for high-speed CNC equipment," Int. J. Adv. Manuf. Technol., 2012.
[36] L. Di Angelo, P. Di Stefano, and E. Guardiani, "A Build-Time Estimator for Additive Manufactured Objects BT - Design Tools and Methods in Industrial Engineering," 2020, pp. 925–935.