*STP 1637, 2022 / available online at www.astm.org / doi: 10.1520/STP163720210013*

Sebastian Dirks,[1] Andreas Collet,[1] and Johannes H. Schleifenbaum[1]

# Open Vector Format for Laser-Based Computer-Aided Manufacturing

**Citation**

S. Dirks, A. Collet, and J. H. Schleifenbaum, "Open Vector Format for Laser-Based Computer-Aided Manufacturing," in *Progress in Additive Manufacturing 2020*, ed. N. Shamsaei and M. Seifi (West Conshohocken, PA: ASTM International, 2022), 117–130. http://doi.org/10.1520/STP163720210013[2]

**ABSTRACT**

Data preparation for additive manufacturing (AM) is a complex process that gained importance as the industrialization of various AM technologies progressed. Lots of effort has been spent on the analysis of three-dimensional (3D) computer-aided design data handling, where the Standard Tessellation Language, or STL, forms a de facto industry standard. To manufacture a given geometry using AM, the 3D data need to be sliced into 2.5D layer-wise data and subsequently into vectors. For AM machines based on multiaxis systems, the G-Code Standard is common for the transfer of already sliced information. Yet, for systems using a galvanometer scanner to change the position of the focus and working point (e.g., laser powder bed fusion), G-Code has several limitations. Proprietary formats are almost exclusively used instead of standards. In this paper, the requirements for a more flexible, open, and technology-independent data format are discussed and analyzed. The data structure, objects, and metadata are deducted from the use case technology laser powder bed fusion. Based on this, an open-source reference implementation of a new format prototype, the Open Vector Format using Google Protocol Buffers technology, is

Manuscript received January 20, 2021; accepted for publication March 25, 2021.

[1]Digital Additive Production, RWTH Aachen University, Aachen, North-Rhine Westphalia, Campus Blvd. 73, 52074, Germany S. D. https://orcid.org/0000-0003-4987-4983, A. C. https://orcid.org/0000-0003-3976-8669, J. H. S. https://orcid.org/0000-0002-7675-6547

[2]ASTM *International Conference on Additive Manufacturing* held virtually on November 16–20, 2020.

presented and its performance is evaluated in research projects of RWTH Aachen University.

**Keywords**
additive manufacturing, file format, computer-aided manufacturing, laser powder bed fusion, layer data, metadata, vector block data, data model

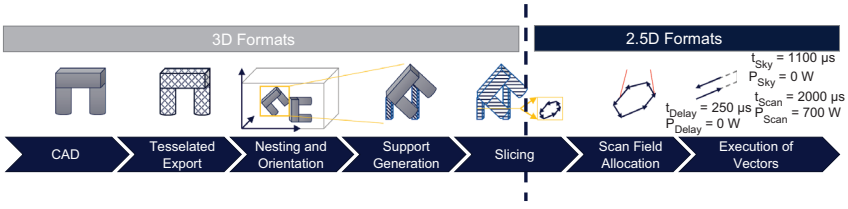
## Introduction

Data preparation for additive manufacturing (AM) is a complex process that has increased in importance together with the industrialization of the technology itself. Many researchers have analyzed the topic of handling three-dimensional (3D) computer-aided design (CAD) data, where the Standard Tessellation Language, or STL, forms a de facto industry standard.[1]

To manufacture a given geometry additively, the 3D data need to be sliced into 2.5D layer-wise data, which is the focus of this paper. Laser powder bed fusion (LPBF) is used as an example of laser-based systems. The characteristics of the LPBF process include using a galvanometer scanner to change the position of the focus and working point. The typical digital process chain of LPBF data preparation is shown in **figure 1**.

Complex 3D objects in CAD kernels are commonly represented using a boundary geometry representation.[2,3] This mathematical representation needs to be discretized and is transformed into triangles (e.g., STL).[4,5] These will be manipulated with AM specialized software such as Materialize Magics, Autodesk Netfabb, or proprietary software of AM system providers such as Concept Laser, EOS, SLM Solutions, Renishaw, 3D Systems, and Trumpf. The parts are placed into the building volume ("nesting") with a defined orientation and supports. After the supports are generated, the parts are sliced into a layer-wise 2.5D representation. Vectors represent the contours and filling volumes (hatches) of the parts. The vectors then are postprocessed in one or more optional steps before they are executed by the machine controller software. An example of a postprocessing step is the allocation of vectors to laser scanner units in a multilaser machine.[6]

**FIG. 1** Data preparation for laser powder bed fusion.



In the machine controller software, the polylines and vectors are transformed into microvector commands as real-time setpoints for the scan system closed loop control. Many state-of-the-art examples can be found in the literature.[7–9]

With each discretization in this data pipeline (tessellation, slicing, microvectors), information of the exact geometry is lost, and the quantity of stored data is increased. In return, both the required computation complexity and the implementation effort are lowered.

Looking at the data formats used to transfer the data along this pipeline, the G-Code Standard is common for AM machines based on multiaxis systems. In LPBF, instead of a data format standard, closed proprietary formats created by the original equipment manufacturers (OEMs) are used almost exclusively. A large portion of scientific literature addressing the topic of AM data formats focuses on 3D geometry formats. Only a few examples can be found that review vector-based 2.5D formats. To the best of the authors' knowledge, no sources can be found that address the topic of making the 2.5D data easily available for use cases other than the print job itself, which is discussed in the section on use cases.

In computer science, general purpose formats are often used to serialize structured data for saving (e.g., as a file) or for transmitting them over a network connection. Serialization refers to the process of copying data from, for example, data objects of a programming language into a series of bytes. All formats require serialization for these operations; in proprietary AM formats, the serialization is handled by proprietary software. There are different technologies and libraries providing serialization functions; some well-known examples are the eXtensible Markup Language (or XML) and JavaScript Object Notation (or JSON). Both use text-based encoding. Thus, they translate floating point numbers and integers into text characters.[10,11] Other technologies such as Protocol Buffers (Protobuf)[12] use a binary encoding that leaves numbers in their native binary representation. In this paper, the general purpose formats are examined as a possible solution for AM data transfer, and the performance of binary and text-based formats is compared.

Research in the area of process quality monitoring[13–15] and process acceleration[16–18] show that LPBF manufacturing technology has potential for improvement in two main areas: productivity and quality of the final product.[19] In the next section, use case examples are presented that show the potential for improving one or both of these process performance indicators, but these are hindered by limitations resulting from the data formats used to transfer 2.5D data. Requirements for an open data transfer are deducted from the use cases. Then, a critical review of the requirements is performed, and three potential solutions are analyzed: the use of an existing standard, opening the proprietary formats, and using a flexible general-purpose structured data format.

In the "Basic Object Model" section, the new Open Vector Format (OVF) is presented as a proposed blueprint solution, followed by a short performance example. OVF is a prototype implementation of the data model derived from the requirements tested on LPBF lab machines at RWTH Aachen University.

## Use Cases

Based on experiences from research projects at RWTH Aachen University, four use cases for an open format are analyzed in this paper. Motivation for each use case is the potential to improve either the LPBF productivity or quality or both. Each use case results in a list of requirements for the data format. The relationship between use cases and requirements is summarized in **table 1**.

There are two requirements resulting from the LPBF process itself: The format must represent ***two-dimensional vectors plane-wise*** in three-dimensional space. The data must be processable by a ***real time-capable*** deflection control system. Thus, it needs to be read fast and with low computational effort in the machine controller software.

1. *Integrating vector data into simulations of the process*: A precise, system-independent calculation of process time needs an open 2.5D data input.[20] These simulations have the potential to optimize productivity by calculating the potential build time reduction of process innovations a priori. Thermal process simulations could use the vector input for calculating local energy input (e.g., simulation for distortion due to the LPBF process requires the consideration of scan strategies and movement).[21–23]

   The integration effort in common programming languages should be low for these use cases. They require metadata (e.g., part relationships of the vectors). The data volume and computation effort should be minimal to minimize the impact on the simulation performance.

**TABLE 1** Requirements for a new 2.5D format

| Requirement | Linked Use Cases | Evaluation of Existing Formats Supporting 2.5D Representation | | |
|---|---|---|---|---|
| | | CLI | G-Code | 3MF (Slice Extension) |
| Two-dimensional vectors plane wise | 0 | Fulfilled | Fulfilled | Fulfilled |
| Real time capable; read fast and with low computational effort | 0, 1, 3 | Fulfilled | Not for scanner-based systems | Fulfilled |
| Low integration effort in common programming languages | 1, 2, 3 | Not fulfilled no libraries or official specification | Not fulfilled | Fulfilled Libraries available |
| Meta data | 1 ,2, 3 | Not fulfilled | Not fulfilled | Fulfilled |
| Minimal data volume | 1, 3 | Fulfilled | Not fulfilled | Not fulfilled |
| Machine parameters | 3, 4 | Not fulfilled | Fulfilled | Not fulfilled |
| Easily extendable | 3, 4 | Not fulfilled | Not fulfilled | Fulfilled |
| Additional primitives | 4 | Not fulfilled | Fulfilled | Not fulfilled |

*Note*: 0 = LPBF process; 1 = process simulation; 2 = process monitoring; 3 = data pipeline software components; 4 = full functionality closed loop control.

2. *Process monitoring based on external devices for the process*: With detailed information about laser vectors, including laser delays and inertia compensation ("sky writing"), a connection between measured sensor information and prepared control commands can be achieved to analyze and thus improve process quality. The ultimate goal of a "real-time, closed-loop feedback control of the additive process"[24] can be supported with precise input data of control commands. Like the simulations, low integration and computation effort and metadata are needed.
3. *New and innovative slicing data pipeline software components*: For both productivity and quality improvements, steps in the slicing data pipeline have the potential for improvement. Some examples include the allocation of vectors to scanners, narrow path optimizations, and local thermal optimization by either lowering energy input or increasing execution speed and thus productivity.[17,18,25,26]

   Innovation speed can be increased when each of these problems can be solved in a dedicated component, not only by the OEM but also by researchers and third parties, for example. All components are finally integrated into a data preparation pipeline, and the resulting overall system is tested.

   This use case results in several requirements: data volume and computation time should be minimal; all possible machine parameters need to be settable within layers (in-between vectors); low integration effort and metadata are needed; and finally, the data need to be structured and easily extendable to cover new use cases.
4. *Utilize full functionality of state-of-the-art laser closed loop controls*: A discretization of the geometric representation as close as possible to the actual machine process control is beneficial for reducing the amount of data transferring and processing in previous steps. The implementation and testing complexity and real-time capabilities of the machine controller limit this possibility. In state-of-the-art scanner control systems, the processing of basic geometric forms is implemented: (poly-)lines, arcs, and ellipses.[7–9]

   Thus, state-of-the-art systems enable the use of these primitives directly at the machine control software level. It is also possible to change various process parameters (e.g., laser power, mark speed, delays) for every primitive. But these capabilities are underutilized in today's digital process chain. The approximation of the boundary geometry representation models with polygons during tessellation causes loss of precision at curved elements, which can be improved by using arcs and ellipses[27] (e.g., for the manufacturing of lattice structures).

   This use case requires support for the additional primitives: (poly-)lines, arcs, and ellipses, as well as extendable machine parameters.

## Critical Requirements Review and Solution Analysis

In current AM software, all the proprietary formats are already implemented. From an OEM's perspective, two considerations naturally arise: the efforts needed to implement a new solution and the protection of the OEM's intellectual property. In this section, three potential solutions to these considerations are analyzed.

### USING EXISTING STANDARDS

The first solution is the use of already existing standards. The requirements are compared to the capabilities of the state-of-the-art formats G-Code,[28] CLI,[29] and 3MF[30] and summarized in table 1. The following shortcomings can be deduced:

1. As a text-based representation, G-Code is inefficient in storing a large number of primitives' coordinates.[31] Due to the typically small size of the laser spot (between 80 and 1000 µm),[32] a complex standard use case (several $dm^3$) is represented by several hundred million point coordinates. Using text-based representation results in data in the range of gigabytes.
2. CLI and 3MF lack support for arc or ellipse primitives.
3. G-Code and CLI are missing parameter definitions for AM manufacturing. CLI has no parameters at all.

Additionally, the integration effort also includes the internal implementation and testing of generating the data according to the standard. The standards are not easily extendable and need time to include changes.

### OPENING THE PROPRIETARY FORMATS TO THE PUBLIC

The approach of opening the proprietary formats lowers the integration effort for the OEM. But every component that uses a different proprietary format needs a software adapter to translate the data between formats. There are two solutions for implementing the adapters: writing one adapter for every pair of the $n$ formats, which results in implementing $n^2$-$n$ adapters, or writing one adapter for a common interchange format, which results in $n$ adapters. The second solution is equal or superior when there are more than two proprietary formats that should be supported. The interchange format needs to support the superset of all format capabilities, which synergizes well with the third solution, using a general-purpose format.

### USING A GENERAL-PURPOSE DATA FORMAT

As presented in the introduction, a general-purpose data format brings the advantages of easy integration in programming languages. To fulfill the performance requirements and gain flexibility, the open-source technology Protocol Buffers has been chosen for the showcase implementation in this paper, and it is reviewed in the "Discussion of Performance" section. The main technical differences of Protocol Buffers compared to other serialization libraries are the use of a binary format for numerical values, binary message tags to enable forward and backward

compatibility of versions, and a code-generation infrastructure to generate native classes in various programming languages. The message structure enables the streaming capability of objects for data pipelines.[12]

### INTELLECTUAL PROPERTY AND INNOVATION

Protecting intellectual property is a motivation for OEMs that contradicts efforts for open formats. Developments in computer aided manufacturing software for computer numerical controls show the potential of an open data ecosystem in conventional multiaxis system data preparation. Software providers utilize the G-Code or STEP-NC standard for innovations. They build libraries and software modules to support the actual computer numerical control system providers with digital processes such as custom programs for performance verification, radio-frequency identification supported program and resource verification, path simulations and visualization, and path collision warning.[31,33] Laser-based AM has the same potential for an interchangeable data pipeline.

The discussed requirements, linked use cases, and the evaluation of G-Code,[28] CLI,[29] and 3MF[30] are visualized in **table 1**.

## Basic Object Model

From the requirements, the OVF data model has been derived and implemented. The basic object model is derived from the layer-wise build process of AM processes. The most basic data object types and their relationships can be found in **figure 2**. A detailed unified modeling language class diagram of all data fields can be found in **figure 3**.

FIG. 2 Basic object model of open vector format.

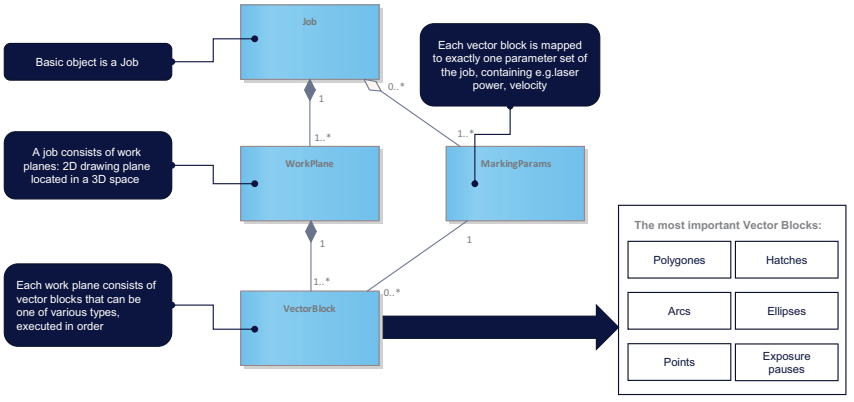
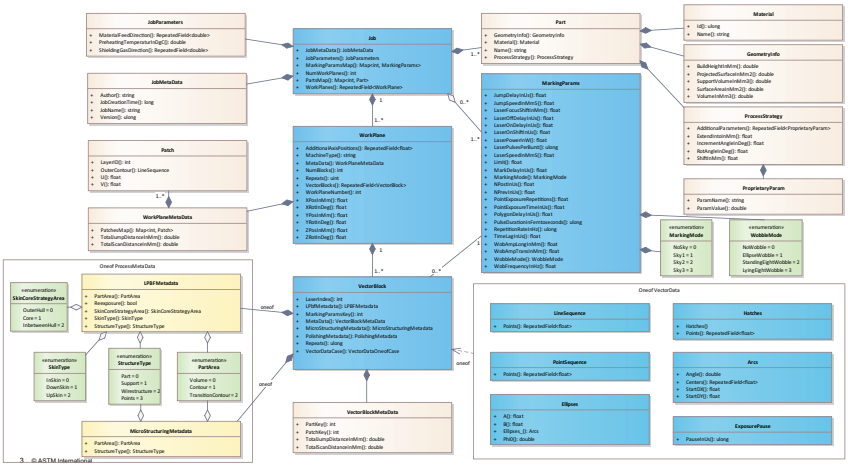


**FIG. 3** OVF data model.




## JOB

The job object is the root object of the model and represents the whole build job the AM machine will manufacture. Contained in the job are two data objects: a list of WorkPlane objects and a map of MarkingParams objects. These data objects contain all the information that is required for the machine to manufacture the job. Additionally, the job contains several metadata objects that are detailed later in this paper. The most important of these metadata objects is the Part object map.

## WORKPLANE

The WorkPlane object represents a two-dimensional drawing plane located in 3D space; its name derives from the CAD concept with a similar function. The WorkPlane is the most fundamental object because it shows and limits the use case of the OVF. The object model is aimed at AM processes that use a motion system (e.g., multiaxis system, industrial robot) to position a build object (e.g., LPBF build plate, laser metal deposition part) relative to a vector-based actuator (e.g., galvanometer scanner with laser source, magnetic field deflector with electron beam). The movements of both actuators are expected to happen consecutively and not simultaneously. Thus, in the example of LPBF, the *z*-axis moves the build plate to the correct height first. After the movement is finished, the laser actuator starts vector execution. Goal coordinates for the motion system are stored in the WorkPlane object itself.

### VECTORBLOCK

The VectorBlock object represents a set of vectors that share the same metadata and process parameter set. Each VectorBlock is mapped to one specific set of marking parameters from the job. The VectorBlock object exists mainly for performance reasons and storage space efficiency. VectorBlocks can adapt to one of several available types that define how exactly the point coordinate data are stored and will be interpreted by the machine controller software. The most common VectorBlock types are Hatches and LineSequences (also known as polygons) that can also be found in CLI.[29]

Hatches are defined as a set of straight lines that are not connected to each other and are commonly used to create the "filling" of parts. LineSequences are a set of connected straight line segments that are commonly used to create contours of AM parts. VectorBlock types act as flexible extension points of the format. The machine controller software will check the vector block type and only execute known types. If a specialized vector block type is beneficial for a use case, both the vector generating code as well as the machine controller can be extended to support other types while still utilizing the other parts of the job structure. For example, arcs and ellipses have been added as additional VectorBlock types for use in Case 4.

Additionally, for multilaser machines, each VectorBlock can store the index of the executing laser unit if the vector allocation is generated upstream and not by the machine controller software.

### MARKINGPARAMS

The MarkingParams object contains process parameters for the execution of the vectors such as marking speed, power, and abstract parameters for configuring the closed loop controller that controls the laser beam's positional accuracy. This includes feed-forward parameters such as *delays* as well as acceleration and deceleration compensation ("skywriting") set points such as *prerun* and *postrun* times. It is up to the machine controller software to calculate the real set points from the abstract description considering the hardware components specifications, safety limits, and so on.

### METADATA

The basic objects Job, WorkPlane, and VectorBlock contain additional fields for metadata. Details can be found in **figure 3**. Most relevant is the *part* object. Many AM processes like LPBF create a whole batch of parts at once. Separation of parts from the build plate is a postprocessing step (e.g., by a saw). The information about the part relationship is not relevant for the printing process itself but is highly relevant for applications that process the data after slicing. Quality-assurance applications (Use Case 2) especially need this relationship to connect data that are

generated during or after AM processing, such as process monitoring sensor data and quality control measurements, with the machine code. The metadata object can be one of various extendable types to support specific requirements of different AM processes, for example, microstructuring (given as example in **fig. 3**) or laser metal deposition.

## Discussion of Performance

The main advantage of the binary data and generated source code is the near-native performance. The focus of this paper is not the exact performance comparison of serialized data formats but the use case of AM data handling. Therefore, the example presented uses AM vector data. A few examples of general serialization performance benchmarks can be found in the literature.[34–37] In general, binary formats perform better in space requirement, serialization, and deserialization time for numerical data, and AM data mostly consists of floating point values for the vector coordinates. Therefore, the expectation for typical AM data is a major performance increase. The exact increase is highly data dependent on the fraction of nonnumerical data as well as the implementation in the target programming language, for example. The following example is therefore only intended to give a rough performance estimate because performance is only one advantage of the technical approach.

The example comparison uses the 3MF file format, which is based on XML. Two proprietary format files have been read in and the contained vector data have been saved—first as OVF messages and a lookup table to enable random access of each WorkPlane. The implementation of the write function can be found in the Git Hub repository.[38] The second file type uses the official 3MF library (lib3mf Version 2.0.0)][30] to write files according to 3MF slice extension specification; 3MF is an XML-based format. The XML files are compressed by lib3mf to save disk space. The expectation for a compression is a decrease in the space requirement with the cost of longer save times.

To compare compressed and uncompressed sizes, the 3mf file has been unpacked and the proprietary and OVF files have been compressed using the fastes" preset of "deflate" ZIP compression by the software 7-Zip file manager (V19.00)[39] with a 32-kB dictionary size and 32-bit word size. Note that the default save format for 3mf is compressed and, for proprietary and OVF, it is uncompressed. The comparison of the sizes can be found in **figure 4**.

As seen in the comparison, the XML-based 3MF files are about five times as big as the same data saved as OVF. This ratio is improved by the 3mf library by compressing the resulting file, which is still slightly larger than the uncompressed OVF, but this increases the runtime of the save operation. The overall save time when using 3mf is about 37 times as high as using OVF, as seen in **figure 5**.

FIG. 4 File-size benchmark of different formats.

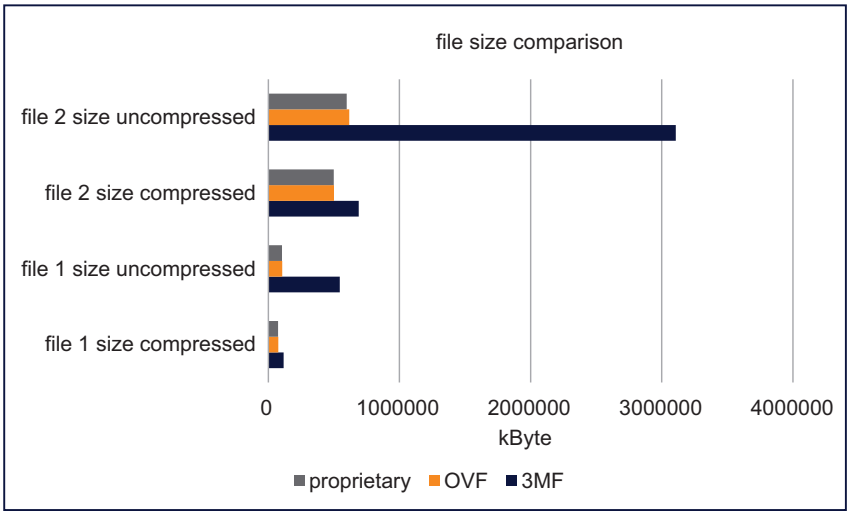
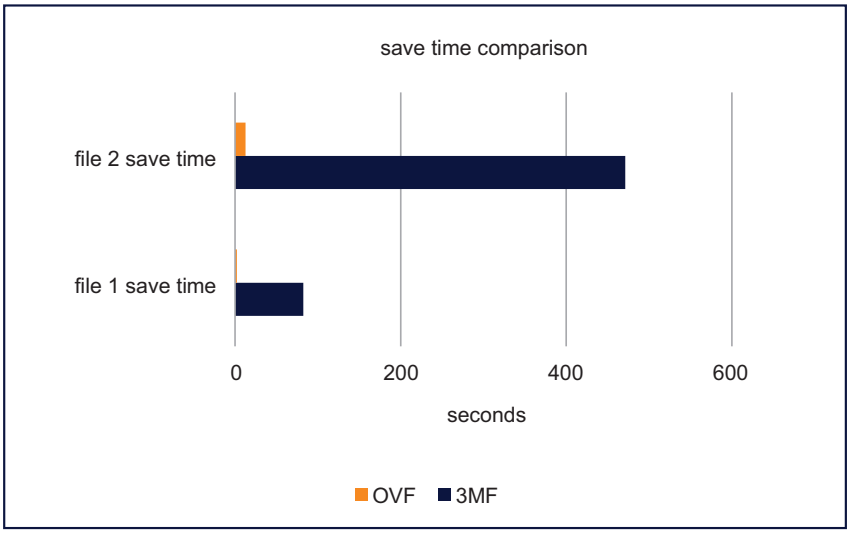



FIG. 5 Save-time benchmark of different formats.




- saving call only was measured in debug mode
- For 3MF official C# 3MFLib. dll was used
- Hardware: AMD Ryzen PRO 2500u Win10
- Toshiba KXG50ZNV 512GB PCIE SSD

## Conclusion

In this work, OVF is presented as a blueprint solution for an AM data model along the process chain. The use cases for an open vector-based format are explained: process simulations, vector postprocessing, process monitoring, and interchangeable data pipelines. The requirements of these use cases are compared to G-Code, CLI, and proprietary formats, and shortcomings are derived: inefficient storage of numerical values, lack of geometric primitives (such as arcs and ellipses that state-of-the-art scanners support), as well as missing process parameters and metadata.

A possible solution in form of OVF is presented. The basic object model consisting of Job, WorkPlane, VectorBlock, and Parameter objects is detailed. The technical fundament using Protocol Buffers—its features and advantages—is introduced. The format performance is compared to the text-based 3mf format in an example.

**ACKNOWLEDGMENTS**

The authors would like to thank the Federal Ministry of Education and Research of Germany for funding the research in the project, Industrialization and Digitalization of Additive Manufacturing for Automobile Series Production (IDAM, grant number 13N15084).

## References

1. M. J. Pratt, A. D. Bhatt, D. Dutta, K. W. Lyons, L. Patil, and R. D. Sriram, "Progress towards an International Standard for Data Transfer in Rapid Prototyping and Layered Manufacturing," *Computer-Aided Design* 34, no. 14 (2002): 1111–1121.
2. W. Giloi, *Interactive Computer Graphics: Data Structures, Algorithms, Languages* (Englewood Cliffs, NJ: Prentice-Hall, 1978).
3. J. Encarnacao, *Computer Aided Design Modelling, Systems Engineering, CAD-Systems* (Berlin, Germany: Springer, 1980).
4. M. Segal, and K. Akeley, "The OpenGL Graphics System: A Specification: Version 4.0 (Core Profile)—March 11, 2010," 2010, https://web.archive.org/web/20210310214740/https://www.khronos.org/registry/OpenGL/specs/gl/glspec40.core.pdf
5. E. Nasr, A. Al-Ahmari, and K. Moiduddin, "CAD Issues in Additive Manufacturing," in *Comprehensive Materials Processing*, S. Hashmi, ed. (Burlington, MA: Elsevier Science, 2014), 375–399.
6. Additive Industries B.V., "Honorable Mention for Additive Industries for Dynamic Multi Laser Assignment!" 2018, https://web.archive.org/web/20201025074034/https://www.additiveindustries.com/news/news-and-press/honorable-mention-for-additive-industries-for-dynamic-multi-laser-assignment
7. SCANLAB GmbH, *Installation und Inbetriebnahme RTC®5 PCI-Karte, RTC®5 PCI-Express-Karte, RTC®5 PC/104-Plus-Karte und RTC®5 PCIe/104-Karte für die Scan-Kopf- und Lasersteuerung in Echtzeit* (Puchheim, Germany: SCANLAB GmbH, 2016), https://web.archive.org/web/20210902092638/https://eu-data.manualslib.com/pdf/de/pdf8/36/3513/351204-scanlab/rtc_5.pdf?f0ae35e9fef2b9f8fc1315ffa58d68b0=&take=binary

8. RAYLASE GmbH, *SP-ICE-3 User's Manual*, (Wessling, Germany: RAYLASE GmbH, 2021), https://web.archive.org/web/20210902090517/https://www.raylase.de/_Resources/Persistent/057e9f84b2fa695c8d7c3d595b9e8da2168fc33d/SP-ICE-3-Users-Manual-v1.44.3.7z
9. Newson NV, *Manual–Application Interfacing with CUA32 Control Units* (Dendermonde, Belgium: Newson NV, 2019), https://web.archive.org/web/20200201215638/http://www.newson.be/doc.php?id=cua32_app
10. *Extensible Markup Language (XML) 1.0 (Fifth Edition)* (Cambridge, MA: World Wide Web Consortium, approved November 26, 2008), https://web.archive.org/web/20210319072210/http://www.w3.org/TR/REC-xml
11. *The JSON Data Interchange Syntax*, 2nd ed. (Geneva, Switzerland: Ecma International, December 2017), https://web.archive.org/web/20210305135903/http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
12. *Protocol Buffers Documentation* (Mountain View, CA: Google, 2020), https://web.archive.org/web/20210312064028/https://developers.google.com/protocol-buffers
13. N. P. Calta, J. Wang, A. M. Kiss, A. A. Martin, P. J. Depond, G. M. Guss, V. Thampy, et al., "An Instrument for In Situ Time-Resolved X-Ray Imaging and Diffraction of Laser Powder Bed Fusion Additive Manufacturing Processes," *Review of Scientific Instruments* 89, no. 5 (2018): https://doi.org/10.1063/1.5017236
14. P. Lott, H. Schleifenbaum, W. Meiners, K. Wissenbach, C. Hinke, and J. Bültmann, "Design of an Optical System for the In Situ Process Monitoring of Selective Laser Melting (SLM)," *Physics Procedia* 12 (2011): 683–690.
15. B. Yuan, G. M. Guss, A. C. Wilson, S. P. Hau-Riege, P. J. DePond, S. McMains, M. J. Matthews, and B. Giera, "Machine-Learning-Based Monitoring of Laser Powder Bed Fusion," *Advanced Materials Technologies* 3, no. 12 (2018), https://doi.org/10.1002/admt.201800136
16. H. Schleifenbaum, W. Meiners, and K. Wissenbach, "Towards Rapid Manufacturing for Series Production: An Ongoing Process Report on Increasing the Build Rate of Selective Laser Melting (SLM)," in *High-Tech Solutions and Concepts*, ed. R. Meyer (Berlin, Germany: Fraunhofer-Allianz Rapid Prototyping, 2008), 71–83.
17. H. Schleifenbaum, A. Diatlov, C. Hinke, J. Bültmann, and H. Voswinckel, "Direct Photonic Production: Towards High Speed Additive Manufacturing of Individualized Goods," *Production Engineering—Research and Development* 5, no. 4 (2011): 359–371.
18. N. Makoana, I. Yadroitsava, H. Möller, and I. Yadroitsev, "Characterization of 17-4PH Single Tracks Produced at Different Parametric Conditions towards Increased Productivity of LPBF Systems—The Effect of Laser Power and Spot Size Upscaling," *Metals* 8, no. 7 (2018): 475–488.
19. *Additive Manufacturing Processes: Quality Grades for Additive Manufacturing of Polymer Parts*, VDI 3405, Part 7 (Düsseldorf, Germany: Verein Deutscher Ingenieure, 2019).
20. S. H. G. Dirks and J. H. Schleifenbaum, Adaption of Cost Calculation Methods for Modular Laser-Powder Bed Fusion (L-PBF) Machine Concepts, in *Metal Additive Manufacturing Conference 2019* (Leoben, Austria: Austrian Society for Metallurgy and Materials, 2019), https://doi.org/10.18154/RWTH-2021-06943
21. E. R. Denlinger, M. Gouge, J. Irwin, and P. Michaleris, "Thermomechanical Model Development and In Situ Experimental Validation of the Laser Powder-Bed Fusion Process," *Additive Manufacturing* 16 (2017): 73–80.
22. L. Mugwagwa, D. Dimitrov, S. Matope, and I. Yadroitsev, "Influence of Process Parameters on Residual Stress Related Distortions in Selective Laser Melting," *Procedia Manufacturing* 21 (2018): 92–99.

23. F. Neugebauer, N. Keller, V. Ploshikhin, F. Feuerhahn, and H. Koehler, "Multi Scale FEM Simulation for Distortion Calculation in Additive Manufacturing of Hardening Stainless Steel" (paper presentation, International Workshop on Thermal Forming and Welding Distortion, Bremen, Germany, April 9–10, 2014), https://web.archive.org/web/20210424092123/https://www.researchgate.net/profile/Frederik-Feuerhahn/publication/266652527_Multi_Scale_FEM_Simulation_for_Distortion_Calculation_in_Additive_Manufacturing_of_Hardening_Stainless_Steel/links/5436717f0cf2dc341db309f9/Multi-Scale-FEM-Simulation-for-Distortion-Calculation-in-Additive-Manufacturing-of-Hardening-Stainless-Steel.pdf
24. T. G. Spears and S. A. Gold, "In-Process Sensing in Selective Laser Melting (SLM) Additive Manufacturing," *Integrating Materials and Manufacturing Innovation* 5, no. 1 (2016): 16–40.
25. SLM Solutions Group, *SLM Build Processor User Manual* (Lübeck, Germany: SLM Solutions Group, 2016).
26. Additive Industries, "Additive Industries Accelerates towards Top 3 Position in Metal AM: New Product Launches and Partnerships Announced in Frankfurt," (Eindhoven, The Netherlands: Additive Industries, B. V., 2017).
27. W. Oropallo and L. A. Piegl, "Ten Challenges in 3D Printing," *Engineering with Computers* 32, no. 1 (2016): 135–148.
28. *Automation Systems and Integration—Numerical Control of Machines: Program Format and Definitions of Address Words*, ISO 6983-1:2009. (Geneva, Switzerland: International Organization for Standardization, 2009).
29. CLI Development Group, "CLI Format Specification," 1994, https://web.archive.org/web/19970617041930/http://www.cranfield.ac.uk/aero/rapid/CLI/cli_v20.html
30. 3MF Consortium, "3MF Specification," 2020, https://web.archive.org/web/20210323112930/https://3mf.io/specification
31. J. Sääski, T. Salonen, and J. Paro, *Integration of CAD, CAM and NC with Step-NC* (Espoo, Finland: VTT Technical Research Centre of Finland, 2005).
32. M. C. Sow, T. de Terris, O. Castelnau, Z. Hamouche, F. Coste, R. Fabbro, and P. Peyre, "Influence of Beam Diameter on Laser Powder Bed Fusion (L-PBF) Process," *Additive Manufacturing* 36 (2020), https://doi.org/10.1016/j.addma.2020.101532
33. ModuleWorks, *5-Axis Component* (Aachen, Germany: ModuleWorks, 2020).
34. S. Bittl, A. A. Gonzalez, M. Spähn, and W. A. Heidrich, "Performance Comparison of Data Serialization Schemes for ETSI ITS Car-to-X Communication Systems," *International Journal on Advances in Telecommunications* 8, nos. 1–2 (2015): 48–58.
35. D. P. Proos and N. Carlsson, "Performance Comparison of Messaging Protocols and Serialization Formats for Digital Twins in IoV" (paper presentation, 2020 IFIP Networking Conference (Networking), Paris, France, June 22–26, 2020).
36. S. Shetye, "SerializersCompare," 2014, https://web.archive.org/web/20210323113000/https://github.com/sidshetye/SerializersCompare
37. E. Smith, "jvm-serializers," 2015, https://web.archive.org/web/20210323113026/https://github.com/eishay/jvm-serializers/wiki
38. Digital-Production-Aachen, "Open Vector Format GitHub Repository," 2021, https://web.archive.org/web/20210323113048/https://github.com/Digital-Production-Aachen/OpenVectorFormat
39. Igor Pavlov, "7-Zip File Manager," 2019, https://web.archive.org/web/20210323113126/https://www.7-zip.org/