

Overhang control in topology optimization

a comparison of continuous front propagation-based and discrete layer-by-layer overhang control

van de Ven, Emiel; Maas, Robert; Ayas, Can; Langelaar, Matthijs; van Keulen, Fred

DOI

[10.1007/s00158-021-02887-2](https://doi.org/10.1007/s00158-021-02887-2)

Publication date

2021

Document Version

Final published version

Published in

Structural and Multidisciplinary Optimization

Citation (APA)

van de Ven, E., Maas, R., Ayas, C., Langelaar, M., & van Keulen, F. (2021). Overhang control in topology optimization: a comparison of continuous front propagation-based and discrete layer-by-layer overhang control. *Structural and Multidisciplinary Optimization*, 64(2), 761-778. <https://doi.org/10.1007/s00158-021-02887-2>

Important note

To cite this publication, please use the final published version (if applicable). Please check the document version above.

Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights. We will remove access to the work immediately and investigate your claim.



Overhang control in topology optimization: a comparison of continuous front propagation-based and discrete layer-by-layer overhang control

Emiel van de Ven^{1,2} · Robert Maas² · Can Ayas¹ · Matthijs Langelaar¹ · Fred van Keulen¹

Received: 13 August 2020 / Revised: 4 December 2020 / Accepted: 16 February 2021 / Published online: 2 June 2021
© The Author(s) 2021

Abstract

Although additive manufacturing (AM) allows for a large design freedom, there are some manufacturing limitations that have to be taken into consideration. One of the most restricting design rules is the minimum allowable overhang angle. To make topology optimization suitable for AM, several algorithms have been published to enforce a minimum overhang angle. In this work, the layer-by-layer overhang filter proposed by Langelaar (*Struct Multidiscip Optim* 55(3):871–883, 2017), and the continuous, front propagation-based, overhang filter proposed by van de Ven et al. (*Struct Multidiscip Optim* 57(5):2075–2091, 2018) are compared in detail. First, it is shown that the discrete layer-by-layer filter can be formulated in a continuous setting using front propagation. Then, a comparison is made in which the advantages and disadvantages of both methods are highlighted. Finally, the continuous overhang filter is improved by incorporating complementary aspects of the layer-by-layer filter: continuation of the overhang filter and a parameter that had to be user-defined are no longer required. An implementation of the improved continuous overhang filter is provided.

Keywords Topology optimization · Additive manufacturing · Overhang angle · Front propagation

1 Introduction

Additive manufacturing (AM) is widely recognized for its capability to manufacture complex components. As the resulting designs of topology optimization (TO) are frequently geometrically complex, the combination of both methods has received significant interest. This interest is even further increased with the advent of metal AM, which opens the possibility to realize optimal functional components with high strength and toughness.

One of the most active research topics on combining TO and AM is the minimization of support structures

required during manufacturing. Since AM is a layer-by-layer manufacturing process, in the majority of industrially relevant processes each layer requires a certain amount of support from the previously constructed layers, for a variety of reasons. For example, in fused deposition modeling, the support is mainly required because a new layer cannot be printed on air, and needs to be at least partly supported by the previous layer (Jiang et al. 2018). For metal powder bed fusion methods, each layer is built on either the already existing part or metal powder. Heat accumulation can be a major issue as the powder has low conductivity (Sih and Barlow 2004), obstructing the path for heat towards the base plate which acts as a heat sink. Here supports provide heat conduction as well as mechanical support to, among others, prevent distortions (Mercelis and Kruth 2006; Wang et al. 2013; Cloots et al. 2013).

Since the production and removal of supports subsequent to printing is costly and time consuming, design rules have been set up to prevent the need for supports (Thomas 2009; Adam and Zimmer 2014; Kranz et al. 2015). For most AM processes, a critical overhang angle α_{oh} has been defined. The overhang angle is the angle between a down-facing surface and the base plate, as shown in

Responsible Editor: W. H. Zhang

✉ Emiel van de Ven
research@emielvanden.nl

¹ Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology, Mekelweg 2, 2628 CD, Delft, The Netherlands

² Royal Netherlands Aerospace Centre, Anthony Fokkerweg 2, 1059 CM, Amsterdam, The Netherlands

Fig. 1 In order to print a given structure (left), it is discretized into layers (right). The overhang angle α_{oh} is defined as the angle between a down-facing surface and the base plate Ω_0

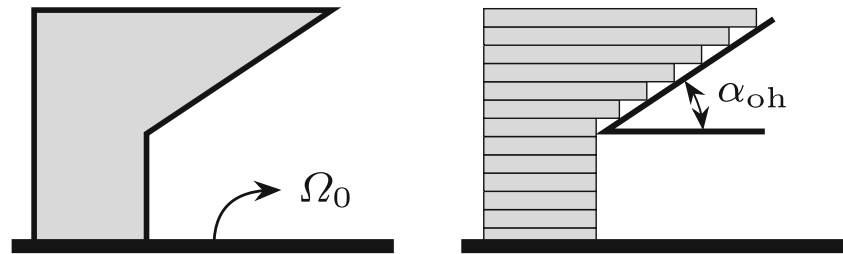


Fig. 1. Surfaces with an overhang angle lower than α_{oh} are termed *overhanging*, and require supports. In order to attain support free optimized structures, numerous studies have proposed various methodologies to incorporate the minimum overhang angle as a design rule in TO. Based on the principle of overhang detection, these can be classified into three categories: overhang detection by (i) a processing of the geometry in printing sequence (e.g. Gaynor and Guest 2016; Langelaar 2016; van de Ven et al. 2018); (ii) inspection of the boundary orientation (e.g. Qian 2017; Guo et al. 2017; Allaire et al. 2017; Zhang et al. 2019); and (iii) simulating the (simplified) physics of the printing process (e.g. Allaire et al. 2017; Amir and Mass 2018; Ranjan et al. 2018). For a comprehensive overview, the reader is referred to Liu et al. (2018).

The focus of this study is on the methods in the first category: the filters that follow the printing sequence. Most methods in this category are presented as discrete filters, defined on a discretized geometry. However, the front propagation-based filter presented in van de Ven et al. (2018) is continuous in nature. This study investigates the differences and similarities between the discrete, layer-by-layer methods presented in Langelaar (2016, 2017), and the continuous, front propagation method presented in van de Ven et al. (2018). Due to similarities in implementation, it was suspected that the discrete layer-by-layer filter could be formulated using front propagation. In this paper it is shown that this is indeed the case. From this, it is concluded that the layer-by-layer filter cannot be used on unstructured grids as is. Furthermore, the front propagation-based filter is improved by using aspects of the layer-by-layer filter, which is enabled by the front propagation-based formulation of the layer-by-layer filter.

This paper is structured as follows. First, a brief overview of both methods is given in Section 2. Then, it is shown that the discrete method can also be formulated with front propagation discretized on a structured grid (Section 3). Next, the differences between both filters are examined (Section 4), and in Section 5 the continuous overhang filter is improved by using aspects of the layer-by-layer filter. Finally, the paper is concluded in Section 6. For clarity this paper focuses on 2D formulations, but the principles apply to the 3D case as well.

2 Overhang detection methods

For the sake of completeness, the overhang detection methods described in Langelaar (2017) and van de Ven et al. (2018) are summarized in this section. In the remainder of this work, the methods described in Langelaar (2017) and van de Ven et al. (2018) will be referred to as the layer-by-layer overhang filter, and the continuous overhang filter, respectively. Both methods are designed for density-based topology optimization, where the geometry is defined by a pseudo-density field ρ where $0 \leq \rho \leq 1$. Void regions are indicated with $\rho = 0$, and regions of material with $\rho = 1$ (Bendsøe and Sigmund 2004). The overhang filter is typically applied directly after the density filter, to convert the density field ρ into a printable density field ξ . The overhang filter removes the overhanging regions: the geometry described by the printable density field ξ is thus directly printable without support structures. This field is then used evaluate the objective and constraints of the topology optimization. Note that the overhang filter can also be implemented as a constraint instead of a filter as demonstrated in van de Ven et al. (2018), which gives the possibility to relax the overhang constraint and balance printability with the impact on performance.

2.1 The layer-by-layer overhang filter

The layer-by-layer overhang filter described in Langelaar (2017) is defined on a structured rectangular grid. In a 2D setting, any element of a structured mesh can be identified by its row and column i, j . The filter can be summarized in two statements: for an element $e_{i,j}$, its supporting elements are the three elements $e_{i-1,j-1}$, $e_{i-1,j}$ and $e_{i-1,j+1}$ directly below it (see Fig. 2), and, the element's printable density

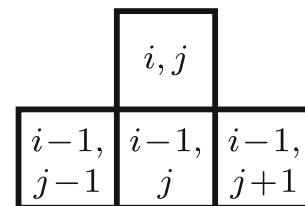


Fig. 2 Element i, j can be supported by the three elements directly below it

$\xi_{i,j}$ cannot exceed the maximum printable density ξ of the elements that support $e_{i,j}$, i.e.

$$\Xi_{i,j} = \max(\xi_{i-1,j-1}, \xi_{i-1,j}, \xi_{i-1,j+1}), \tag{1}$$

$$\xi_{i,j} = \min(\Xi_{i,j}, \rho_{i,j}), \tag{2}$$

where the printing direction is assumed to be from the bottom to the top row. The filter can be propagated through a domain by initializing the bottom row of elements with $\xi_{1,j} = \rho_{1,j}$, and then evaluating (1) and (2) layer-by-layer in the printing direction. Evaluation of sensitivities is performed in the reverse order. For continuous differentiability and the use of gradient-based optimization, the non-smooth min and max operators are replaced by differentiable counterparts.

Note that the overhang angle is inherently linked to the grid. For instance, a square element implies $\alpha_{oh} = \pi/4$. When a suitable structured finite element mesh is used, this filter can thus be applied directly to the elemental densities. In other cases, a mapping between FE mesh and a separate overhang grid must be used (Langelaar 2018). Alternatively, Hoffarth et al. introduce a search cone to define support relations on arbitrary meshes, instead of a mapping (Hoffarth et al. 2017).

2.2 Continuous overhang filter

The overhang filter presented in van de Ven et al. (2018) utilizes front propagation to detect overhanging regions. In a front propagation scheme, a front is initialized at a boundary, and propagated throughout the domain. The

output is an arrival time field $T(\mathbf{x})$, which indicates for each location the time at which the front reaches that location.

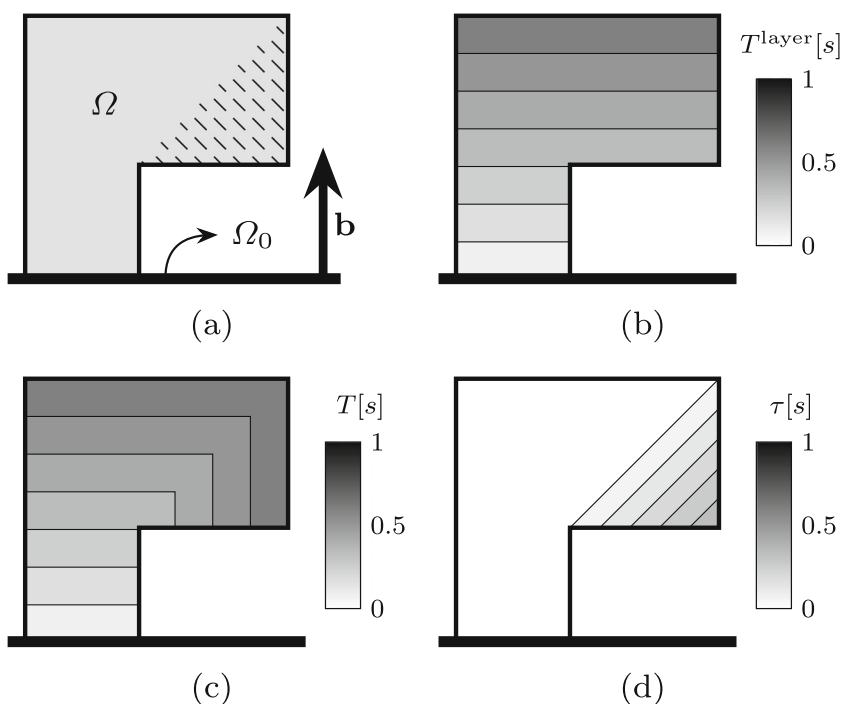
First, the concept of detecting overhang with front propagation is demonstrated on a given geometry, ignoring the dependence on the density field ρ . In order to detect overhang in a given geometry, such as the one shown in Fig. 3a, two arrival time fields are required. In the first arrival time field, T^{layer} , the arrival times are proportional to the height from the base plate Ω_0 , and isosurfaces thus represent the printing sequence, as displayed in Fig. 3b. Assuming that the base plate coincides with the origin, it is given as

$$T^{layer}(\mathbf{x}) = \mathbf{b} \cdot \mathbf{x}/f_0, \tag{3}$$

where \mathbf{b} is a unit vector defining the build direction, and f_0 represents the default propagation speed in this build direction. The exact value of f_0 is not important as it is a scale factor whose effect is canceled, and is chosen as 1 ms^{-1} . For the second arrival time field, $T(\mathbf{x})$, a front propagation is performed in the design domain Ω , starting from the base plate Ω_0 . This time, the propagation speed is chosen such that the arrival times are equal to $T^{layer}(\mathbf{x})$, except when the front travels in a direction *below* the minimum overhang angle α_{oh} , i.e. when a region is not printable. This is achieved using an anisotropic speed function, and the resulting arrival time field can be seen in Fig. 3c. Consequently, a delay field $\tau(\mathbf{x})$ between the two arrival time fields can be calculated as

$$\tau(\mathbf{x}) = T(\mathbf{x}) - T^{layer}(\mathbf{x}). \tag{4}$$

Fig. 3 The shaded overhanging region in the geometry given in (a) can be detected with front propagation by constructing the arrival time fields given in (b) and (c), and evaluating the difference (d). The vector \mathbf{b} indicates the build direction



In regions where $\tau(\mathbf{x}) > 0$, the propagation has been delayed, indicating that the structure is overhanging, as can be seen in Fig. 3d. Finally, the printable density field $\xi(\mathbf{x})$ is attained from the delay field $\tau(\mathbf{x})$ with the delay-density relation h :

$$\xi(\mathbf{x}) = h(\tau(\mathbf{x})), \tag{5}$$

where h is a monotonically decreasing differentiable function with $h(0) = 1$, such that zero delay results in full density, and $0 \leq h(\tau(\mathbf{x})) < 1$ for $\tau(\mathbf{x}) > 0$. In (van de Ven et al. 2018), h is defined as

$$h(\tau(\mathbf{x})) = 2^{-k\tau(\mathbf{x})}, \tag{6}$$

where k is a numerical parameter related to the element size: a high value for k will result in a short transition from full density to void as small delays will already result in low values for ξ . Therefore, for fine meshes, k is set to a higher value than for coarse meshes.

The first arrival time field, $T^{\text{layer}}(\mathbf{x})$, can be obtained without performing an actual front propagation, as the arrival times are equal to the distance between the base plate Ω_0 and the point of interest \mathbf{x} . However, in order to obtain the second arrival time field $T(\mathbf{x})$, a front propagation is performed, which is governed by the Hamilton-Jacobi-Bellman equation:

$$\min_{\mathbf{a} \in S_1} \{(\nabla T(\mathbf{x}) \cdot \mathbf{a}) f_s(\rho(\mathbf{x}), \mathbf{a}, \alpha_{\text{oh}})\} = 1, \quad \mathbf{x} \in \Omega, \tag{7}$$

where f_s anisotropic is the speed function that dictates the propagation speed, and \mathbf{a} is a unit vector determining the direction of propagation: $\mathbf{a} \in S_1, S_1 = \{\mathbf{a} \in \mathbf{R}^2 \mid \|\mathbf{a}\| = 1\}$. It is in the speed function f_s where the original density field ρ is coupled to the arrival times $T(\mathbf{x})$ and thus printable densities $\xi(\mathbf{x})$. It is later shown that the speed function is simply scaled with the densities. Since all the densities are printable at the base plate Ω_0 , it is required that

$$\xi(\mathbf{x}) = \rho(\mathbf{x}), \quad \mathbf{x} \in \Omega_0. \tag{8}$$

From this relation and (5), the initial condition for the front propagation can be derived as

$$T(\mathbf{x}) = h^{-1}(\rho(\mathbf{x})), \quad \mathbf{x} \in \Omega_0, \tag{9}$$

where h^{-1} is the inverse function of the delay-density relation h , such that $h(h^{-1}(\rho(\mathbf{x}))) = \rho(\mathbf{x})$. For example, elements with $\rho = 1$ at the base plate are initialized at $T = T_{\text{layer}}$.

2.2.1 The speed function

For the sake of simplicity, the anisotropic speed function was scaled linearly with the density value ρ . As such, the speed function f_s can be decomposed into a part that depends on the density field, and a part that depends on

the direction of propagation \mathbf{a} and minimum overhang angle α_{oh} :

$$f_s(\rho(\mathbf{x}), \mathbf{a}, \alpha_{\text{oh}}) = g(\rho(\mathbf{x})) f(\mathbf{a}, \alpha_{\text{oh}}). \tag{10}$$

The function $f(\mathbf{a}, \alpha_{\text{oh}})$ relates the direction of propagation to speed. The speed function as used in van de Ven et al. (2018) is displayed in Fig. 4, where α represents the propagation direction defined by \mathbf{a} . The top part of the speed function, where $\alpha_{\text{oh}} \leq \alpha \leq 180^\circ - \alpha_{\text{oh}}$, is defined by $f = f_0 / \sin \alpha$. When the propagation direction is equal to the build direction ($\alpha = 90^\circ$), the propagation speed is equal to that of the reference field T^{layer} , i.e. $f = f_0$. Moreover, in order to maintain a front parallel to the base plate, the propagation speed is increased to compensate for the larger distance travelled to the next printing layer. For example, for $\alpha = 45^\circ$, $f = \sqrt{2} f_0$. However, when the propagation direction is below α_{oh} , i.e. when $\alpha < \alpha_{\text{oh}}$ or $\alpha > 180^\circ - \alpha_{\text{oh}}$, the propagation speed is lowered, thereby a non zero value of $\tau(\mathbf{x})$ is attained. The anisotropic speed function capable of creating a delay in the overhanging regions is not unique. In van de Ven et al. (2018) a function resembling a rectangle is used, which is numerically efficient. The aspect ratio of the rectangle shown in Fig. 4 designates the minimum overhang angle α_{oh} .

Finally, the speed function f_s also contains a geometry dependence in the speed-density relation $g(\rho(\mathbf{x}))$. In the example shown in Fig. 3, the front propagated only through the structure. However, with topology optimization, this structure is determined by the density field ρ . In van de Ven et al. (2018), the propagation speed is linearly scaled with the density:

$$g(\rho(\mathbf{x})) = v_{\text{void}} + (1 - v_{\text{void}})\rho(\mathbf{x}), \tag{11}$$

where v_{void} is the minimum propagation speed in the void regions, chosen such that $0 < v_{\text{void}} < 1$. Consequently, the

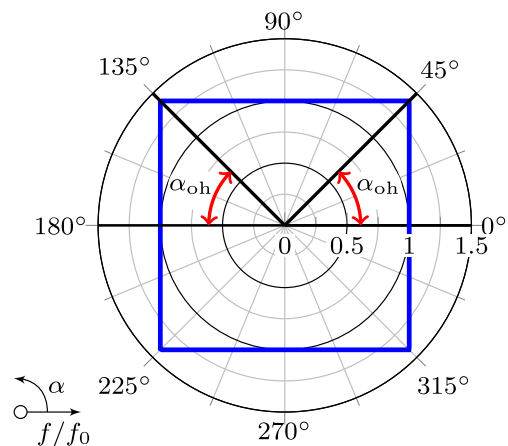


Fig. 4 Polar plot of the rectangular speed function for $\alpha_{\text{oh}} = 45^\circ$, with propagation speed on the radial axis, and propagation direction on the tangential axis. A propagation direction of $\alpha = 90^\circ$ coincides with the build direction \mathbf{b}

front will be delayed in void regions, leading to $\tau(x) > 0$, and thus also void in the printable density field ξ (5).

2.2.2 Numerical implementation

The layer-by-layer overhang filter by Langelaar (2017) is defined on a discretized domain, hence its numerical implementation is straightforward (see Langelaar (2017) for details). Here, the numerical implementation of the continuous filter is briefly described in order to make a comparison between the two.

The front propagation can be efficiently executed using the Ordered Upwind Method (OUM) (Sethian and Vladimirsky 2003), on any mesh type in 2D as well as 3D. The necessary steps are briefly explained, while some details are omitted for brevity. In the OUM, nodes are labelled either *Far*, *Candidate*, or *Accepted*. All of the nodes are initialized as *Far*, with $T = \infty$. Then, the arrival times of nodes at the fixed boundary Ω_0 are initialized and are labelled *Accepted*. Finally, the nodes within a distance d of these *Accepted* nodes are labelled *Candidate*. Then, the following algorithm is executed:

1. Calculate the arrival times of the *Candidate* nodes.
2. Label the *Candidate* node with the lowest arrival time as *Accepted*, and label the nodes within a distance d of that node, that are not *Accepted*, as *Candidate*.
3. If there are any *Candidate* nodes left, go to Step 1.

The distance d is defined as

$$d = zF_2/F_1, \tag{12}$$

where z is a typical element length, and F_1 and F_2 are the maximum and minimum values of $f(\mathbf{a}, \alpha_{oh})$, respectively. Thus, the more anisotropic the speed function, the larger this distance.

A crucial part of the algorithm is the calculation of the arrival times of *Candidate* nodes. Let us first consider the calculation of the arrival time of node i from a given point \mathbf{x}_γ . This is simply the distance divided by the speed, plus the arrival time at point \mathbf{x}_γ , defined as T_γ :

$$T_\gamma^i = \frac{\|\mathbf{x}_i - \mathbf{x}_\gamma\|}{f_s(\rho(\mathbf{x}_i), \mathbf{a}, \alpha_{oh})} + T_\gamma, \tag{13}$$

where $\mathbf{a} = (\mathbf{x}_i - \mathbf{x}_\gamma)/\|\mathbf{x}_i - \mathbf{x}_\gamma\|$, and T_γ^i denotes the arrival time at \mathbf{x}_i when calculated from \mathbf{x}_γ . In reality, the arrival time of the *Candidate* node i is calculated from, in 2D, two *Accepted* nodes, j and k . The point \mathbf{x}_γ is then a point on the segment $\mathbf{x}_j\mathbf{x}_k$, as displayed in Fig. 5. The position and arrival time are linearly interpolated between nodes j and k , defined by parameter γ :

$$\mathbf{x}_\gamma = \mathbf{x}_j(1 - \gamma) + \mathbf{x}_k\gamma, \tag{14}$$

$$T_\gamma = T_j(1 - \gamma) + T_k\gamma. \tag{15}$$

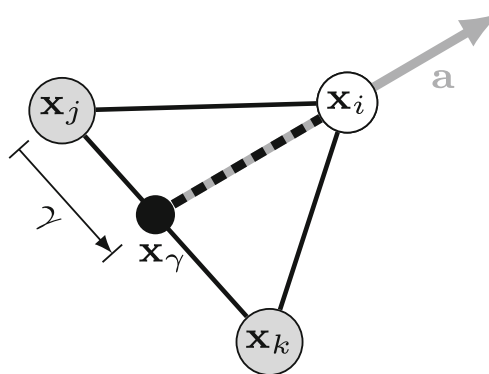


Fig. 5 Update of a node \mathbf{x}_i from \mathbf{x}_γ on the line segment $\mathbf{x}_j\mathbf{x}_k$. The position of \mathbf{x}_γ is determined by $\gamma \in [0, 1]$. The update direction is given by the unit vector \mathbf{a}

The point from which \mathbf{x}_i is updated is the point that results in the lowest arrival time:

$$T_{jk}^i = \min_{\gamma \in [0,1]} T_\gamma^i = \min_{\gamma \in [0,1]} \frac{\|\mathbf{x}_i - \mathbf{x}_\gamma\|}{f_s(\rho(\mathbf{x}_i), \mathbf{a}, \alpha_{oh})} + T_\gamma, \tag{16}$$

where T_{jk}^i denotes the arrival time at \mathbf{x}_i when calculated from the segment $\mathbf{x}_j\mathbf{x}_k$. Note that there is a dependency between the speed function f_s and γ through the propagation direction \mathbf{a} . The final arrival time at \mathbf{x}_i is the minimum arrival time that can be obtained from all the segments between adjacent *Accepted* nodes within a distance d :

$$T_i = \min_{\mathbf{x}_j\mathbf{x}_k \in \text{NF}(\mathbf{x}_i)} T_{jk}^i, \tag{17}$$

where $\text{NF}(\mathbf{x}_i)$ is the set of *Accepted* nodes that are within the distance d of \mathbf{x}_i .

In 3D the algorithm is unchanged, except that the propagation is based on tetrahedrons instead of triangles: the arrival time for a node i is calculated from three other nodes instead of two, where the point \mathbf{x}_γ is now in the triangle spanned by these three nodes instead of a line segment.

Analogous to the layer-by-layer filter, the sensitivities are evaluated by following the reverse order in which the arrival times are evaluated. Since the order of the front propagation is already known, this reduces to a trivial loop over all the elements, as detailed in van de Ven et al. (2018).

3 Formulating the layer-by-layer filter with front propagation

Because the continuous filter is described in a continuous setting, it can be readily used in unstructured meshes while the maximum overhang angle can be adjusted independent of the mesh by simply modifying the speed function. However, in the previous section it became apparent that there is a considerable difference in complexity between

the two methods: the layer-by-layer filter can be described by two equations while the continuous filter is much more involved. This is caused for a large part by the fact that the layer-by-layer filter is a discrete formulation, while the front propagation filter is continuous, which makes it difficult to compare both methods. Therefore, in this section, the layer-by-layer filter is cast into in an equivalent continuous formulation based on front propagation. To this end, the speed function (10) and the speed-density relation (11) have to be changed compared to the continuous filter, which is shown in the following.

3.1 Speed function for the continuous layer-by-layer filter

In order to represent the layer-by-layer filter with front propagation, first, the speed function is adapted. The speed function utilized in van de Ven et al. (2018) allows propagation in all directions (Fig. 4). However, in the layer-by-layer filter, the printable density of an element only depends on either of the three elements directly below it (1). Therefore, the wedge-shaped speed function is chosen as displayed in Fig. 6 (its 3D equivalent is a cone). With this speed function, the propagation speed below the overhang angle is set to zero, and is given by

$$f(\mathbf{a}, \alpha_{oh}) = \begin{cases} \frac{f_0}{\mathbf{b} \cdot \mathbf{a}} & \mathbf{b} \cdot \mathbf{a} \geq \sin(\alpha_{oh}) \\ 0 & \mathbf{b} \cdot \mathbf{a} < \sin(\alpha_{oh}) \end{cases}, \quad (18)$$

where, similar to the previous section, \mathbf{a} and \mathbf{b} are the propagation and build direction, respectively.

This speed function can be propagated on a structured grid with $\Delta y/\Delta x = \tan(\alpha_{oh})$, where Δx and Δy are element width and height, and where the build direction coincides with the vertical axis, as displayed in Fig. 7. Because the

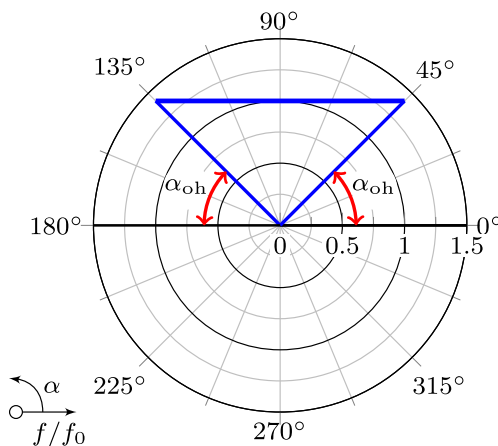


Fig. 6 Polar plot of the wedge-shaped speed function, with propagation speed on the radial axis, and propagation direction on the tangential axis, for $\alpha_{oh} = 45^\circ$. A propagation direction of $\alpha = 90^\circ$ coincides with the build direction \mathbf{b}

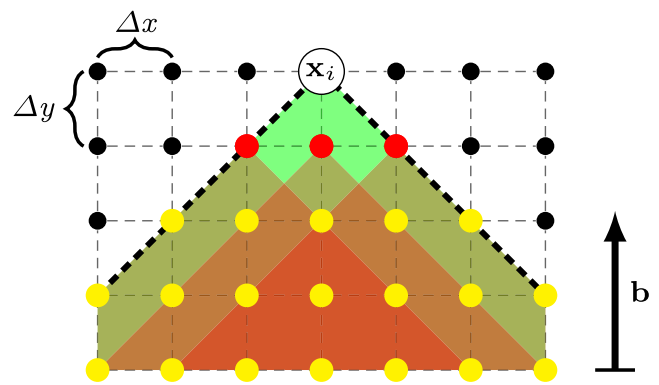


Fig. 7 The node \mathbf{x}_i can be updated from any node in the green area. The red nodes can be updated from nodes in the red area, covering all the yellow nodes. Due to the overlap, \mathbf{x}_i only has to be updated by the red nodes

propagation speed in (18) is zero for propagation directions below α_{oh} , a node can only be updated from nodes in a wedge below it, as indicated with the shaded green region in Fig. 7 where node i can only be updated from the red and yellow nodes. The same holds for the three red nodes directly below \mathbf{x}_i , which can be updated from nodes in the region shaded red. Due to the overlap of the red and green regions, it can be seen that any yellow node that can update node i , can also update at least one of the three red nodes. Since the red nodes are closer to the yellow nodes than node i is, the front will pass the red nodes first. In order to propagate the speed function in Fig. 6, it is therefore sufficient to only check for updates from the three nodes directly below a given node.

With only the three nodes directly below the node of interest required for the arrival time calculation, (17) simplifies to:

$$T_i = \min \{ T_{jk}^i, T_{kl}^i \}, \quad (19)$$

where node i is the node of interest, and nodes j, k , and l are the three nodes below i , as displayed in Fig. 8. This

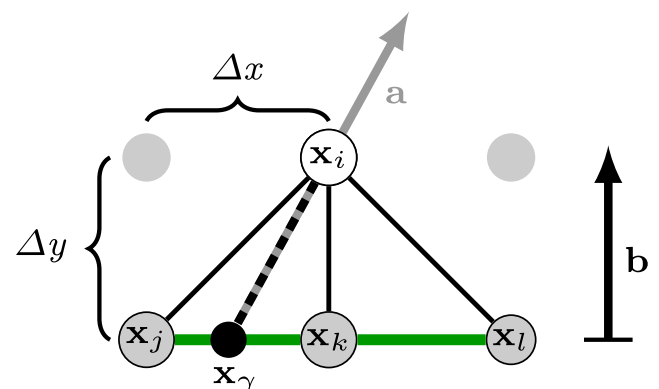


Fig. 8 Update of a node \mathbf{x}_i on a structured grid can occur only from the segments $\mathbf{x}_j\mathbf{x}_k$ and $\mathbf{x}_k\mathbf{x}_l$

means that node i is updated from a point on either $\mathbf{x}_j\mathbf{x}_k$ or $\mathbf{x}_k\mathbf{x}_l$. Since for any position \mathbf{x}_γ on both line segments $\mathbf{b} \cdot \mathbf{a} \geq \sin(\alpha_{oh})$, the speed function in (18) reduces to $f(\mathbf{a}, \alpha_{oh}) = f_0/(\mathbf{b} \cdot \mathbf{a})$. Substituted into (13) this gives

$$T_\gamma^i = \frac{\mathbf{b} \cdot (\mathbf{x}_i - \mathbf{x}_\gamma)}{g(\rho(\mathbf{x}_i))f_0} + T_\gamma. \tag{20}$$

Since the build direction \mathbf{b} is a unit vector and \mathbf{x}_γ moves perpendicular to \mathbf{b} with changing γ , $\mathbf{b} \cdot (\mathbf{x}_i - \mathbf{x}_\gamma) = \Delta y$ (Fig. 8). This implies that the changing distance between \mathbf{x}_i and \mathbf{x}_γ has no influence on the final arrival time. Looking at the shape of the speed function in Fig. 6, this makes sense. As \mathbf{x}_γ moves from \mathbf{x}_k to \mathbf{x}_j in Fig. 8, the distance $\mathbf{x}_i\mathbf{x}_\gamma$ increases, but the speed increases proportionally, as the direction of propagation changes. This reduces (20) to

$$T_\gamma^i = \frac{\Delta y}{g(\rho(\mathbf{x}_i))f_0} + T_\gamma. \tag{21}$$

In order to find the point \mathbf{x}_γ that results in the lowest arrival time T_γ^i , this equation is differentiated w.r.t. γ :

$$\frac{\partial T_\gamma^i}{\partial \gamma} = T_k - T_j. \tag{22}$$

As the right-hand side of (22) is a constant term, the minimum lies at the bounds of each segment. This reduces (19) to

$$T_i = \min \{T_j^i, T_k^i, T_l^i\}, \tag{23}$$

where T_X^i represents the arrival time at node i when calculated from node X , as shown in (21).

3.2 Delay-density and speed-density relation

The change of speed function was the first step to represent the layer-by-layer filter with a front propagation formulation. It can be seen that (23) already resembles (1), where the arrival time value of a node depends on the three nodes below it. The next step is to choose the speed-density relation (11) and the delay-density relation (5) such that the behaviour of the layer-by-layer filter is obtained. For this part, we ignore without loss of generality that each element can be supported by three elements, as this is now inherent to the specific speed function chosen, and we will be looking at single columns of elements only, as depicted in Fig. 9a.

Let us first examine how the printable densities are related to original densities in the layer-by-layer filter. For a single column of elements, such as the one displayed in Fig. 9a, (1) and (2) reduce to

$$\xi_i = \min(\xi_{i-1}, \rho_i), \tag{24}$$

$$= \min(\rho_1, \rho_2, \dots, \rho_i), \tag{25}$$

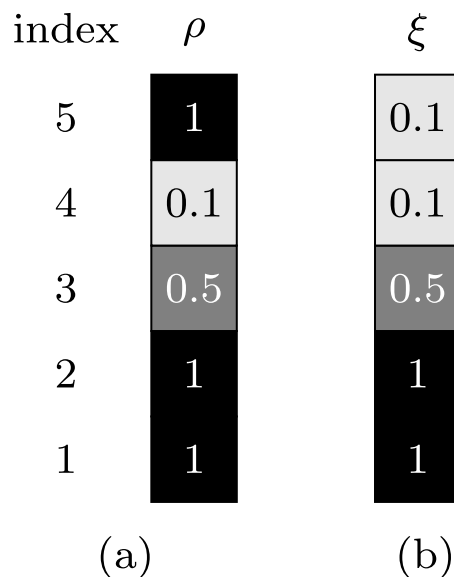


Fig. 9 Demonstration of the conversion of densities ρ a to printable densities ξ b by the conventional layer-by-layer filter on a single column of elements

where i is the index of each element, with $i = 1$ being the bottom element. Thus, the printable density of element i is equal to the lowest density encountered in elements 1 to i . This ensures that an element can only have full printable density if all the elements that support it also have full density. This can be seen in Fig. 9b, where the densities given in Fig. 9a have been processed according to (25). Here, e.g. elements 4 and 5 have $\xi = 0.1$, as $\rho_4 = 0.1$ is the lowest density in the set. The density of element 5 is reduced from 1 to 0.1 as it is not properly supported.

In order to describe the layer-by-layer filter with front propagation, the speed-density and delay-density relations need to be chosen such that (24) is obtained. Recall that with the front propagation formulation, the printable density is defined as (5)

$$\xi(\mathbf{x}) = h(\tau(\mathbf{x})) = h\left(T(\mathbf{x}) - T^{\text{layer}}(\mathbf{x})\right). \tag{26}$$

In a discrete single column setting as in Fig. 9, and substituting (21) for the arrival time $T(\mathbf{x})$, the printable density for an element i (26) reduces to

$$\xi_i = h\left(T_{i-1} + \frac{\Delta y}{g(\rho_i)f_0} - T_i^{\text{layer}}\right). \tag{27}$$

The terms between brackets are the following: the arrival time of the previous element, the distance to the current element divided by the propagation speed, and the arrival time of the reference field. In order to obtain the layer-by-layer filter, the speed-density relation g is chosen such that (24) is obtained from (27):

$$h\left(T_{i-1} + \frac{\Delta y}{g f_0} - T_i^{\text{layer}}\right) = \min(\xi_{i-1}, \rho_i), \tag{28}$$

where the arguments of the speed-density relation g are omitted for brevity. The arrival time of the previous element, T_{i-1} , can be substituted by its printable density using (26):

$$h \left(h^{-1}(\xi_{i-1}) + T_{i-1}^{\text{layer}} + \frac{\Delta y}{g f_0} - T_i^{\text{layer}} \right) = \min(\xi_{i-1}, \rho_i). \quad (29)$$

Using $T_i^{\text{layer}} - T_{i-1}^{\text{layer}} = \Delta y / f_0$ (3), the following speed-density relation is derived:

$$g = \frac{\Delta y}{f_0 (h^{-1}(\min(\xi_{i-1}, \rho_i)) - h^{-1}(\xi_{i-1})) + \Delta y}. \quad (30)$$

This can be interpreted as follows. If $\rho_i > \xi_{i-1}$, the arrival time should be such that $\xi_i = \xi_{i-1}$, or $h(\tau_i) = h(\tau_{i-1})$. In other words, the delay should not increase, which is achieved when the front propagates with the same speed as with which T^{layer} increases over distance, i.e. when $g = 1$. This corresponds to (30), which reduces to 1 when $\rho_i > \xi_{i-1}$.

For $\rho_i < \xi_{i-1}$, the arrival time should ensure that $\xi_i = \rho_i$. Because $\rho_i < \xi_{i-1}$, the delay should increase to lower the printable density ξ_i as compared to ξ_{i-1} . With the speed-density relation given in (30), the speed is lowered exactly such that the new delay will result in $\xi_i = \rho_i$.

3.3 Comparison of front propagation and layer-by-layer formulation

With the speed-density relation in (30) and the speed function given in (18), the layer-by-layer AM filter can be described in a continuous setting with front propagation. For comparison of these formulations, both the layer-by-layer formulation as provided in Langelaar (2017) and its front propagation implementation described in this section are applied to a given density field displayed in Fig. 10a. The resulting printable density fields from the layer-by-layer filter and the front propagation implementation are shown in Fig. 10b and c, respectively. Although they appear visually similar, there are some differences, displayed in Fig. 10d. As can be seen, there is hardly a difference in printable regions ($\xi = 1$), but for intermediate density regions the printable density values differ. This is due to the different minimum approximations that are used. For the front propagation formulation, the minimum approximation are chosen as close as possible to the default settings given in Langelaar (2017). When the smooth minimum/maximum approximation are chosen more accurate and less smooth, the error between both implementations vanishes, and the layer-by-layer filter can be exactly reproduced with front propagation.

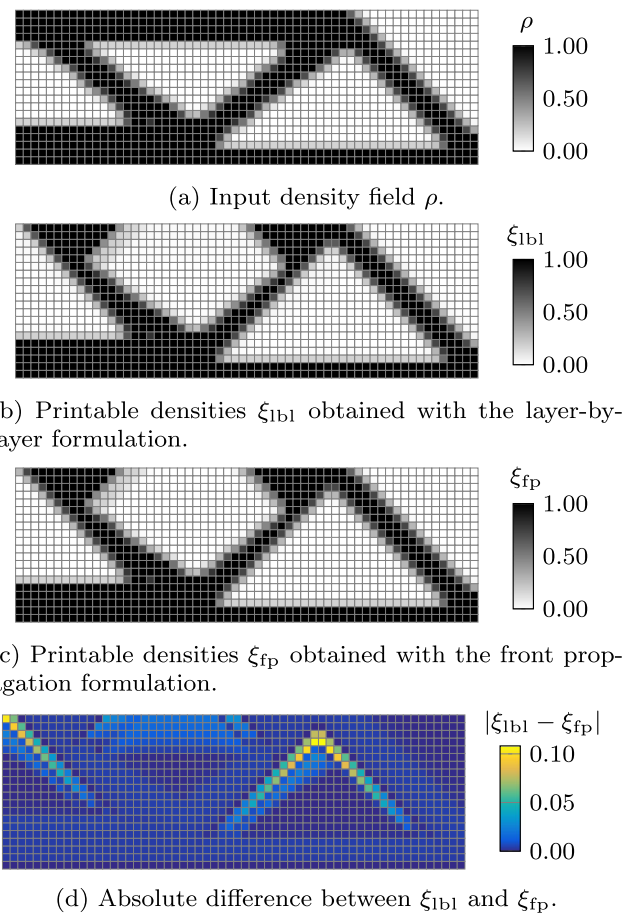


Fig. 10 Application of the layer-by-layer and front propagation formulation on the density field given in (a). The differences occur only in the intermediate density regions, as different minimum approximation are used (d). **a** Input density field ρ . **b** Printable densities ξ_{lbl} obtained with the layer-by-layer formulation. **c** Printable densities ξ_{fp} obtained with the front propagation formulation. **d** Absolute difference between ξ_{lbl} and ξ_{fp}

4 Comparison of layer-by-layer and front propagation-based overhang filters

In the previous section, it is shown that the layer-by-layer filter by Langelaar (2017) can be cast into a front propagation formulation. The front propagation formulation of the layer-by-layer filter enables a detailed comparison between the layer-by-layer overhang filter and the front propagation-based overhang filter presented in van de Ven et al. (2018). To avoid confusion of both methods, the front propagation implementation of the layer-by-layer filter, as shown in Section 3, is referred to as simply the layer-by-layer overhang filter. The original continuous, front propagation based, overhang filter, as introduced in Section 2.2, is referred to as the continuous overhang filter.

The two elements of the front propagation that differ between both filters are summarized in Fig. 11. First, the difference in the speed function $f(\mathbf{a}, \alpha_{\text{oh}})$ is discussed,

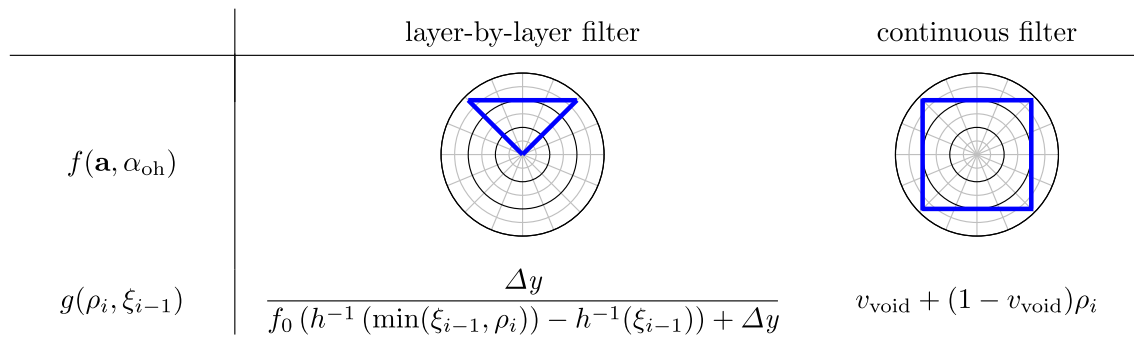


Fig. 11 Differences in speed function and speed-density relation between the layer-by-layer and the continuous overhang filter

followed by the differences in speed-density relation. Finally, the numerical implementations of the various minimum/maximum operators and their effect on the sensitivities are analysed.

4.1 Differences in speed function

The speed functions of both implementations share a similar profile for propagation directions above the minimum allowable overhang angle, i.e. when $\mathbf{b} \cdot \mathbf{a} \geq \sin(\alpha_{oh})$ (18). However, for other directions of propagation, the speed functions are different, as can be seen in Fig. 11 (both speed functions are displayed in more detail in Figs. 4 and 6). The rectangular speed function of the continuous filter allows propagation in all directions, even opposite to the build direction, and is only slightly anisotropic, with an anisotropy ratio, i.e. maximum speed divided by the minimum speed, of $\sqrt{2}$ for $\alpha_{oh} = 45^\circ$. The triangular speed function of the layer-by-layer filter on the other hand, only allows propagation when $\mathbf{b} \cdot \mathbf{a} \geq \sin(\alpha_{oh})$ while the propagation speed is zero for other directions. It therefore has an infinite anisotropy ratio.

The layer-by-layer speed function has a close resemblance to the AM process, as it can only propagate in directions that are printable from its current position. It is therefore a more natural choice, and was investigated for the continuous filter as well, before adopting the rectangular speed function for numerical reasons further detailed in the following.

For practical applicability, one of the requirements for the continuous filter was that it should be applicable to unstructured meshes. On unstructured meshes, it is difficult to propagate a front with a highly anisotropic speed function. As mentioned in Section 2.2, a crucial part of the front propagation algorithm is the calculation of the arrival time for a node when updated from two other nodes. This update is executed several times per node, and is therefore the dominant factor in computational cost. Reducing the number of updates greatly reduces the computational cost.

The number of updates required per node depends on the distance d from which a node can be updated, which in turn depends on the anisotropy ratio of the speed function (12). Ideally, one would update a node only from its direct neighbours, as is the case for an isotropic speed function. However, for the speed function of the layer-by-layer filter, as the minimum speed equals zero, $d = \infty$, meaning a node can be updated from every other node in the mesh. This is demonstrated on the unstructured mesh given in Fig. 12a. Here, a front is propagated from the blue node with the wedge-shaped speed function as given in Fig. 6. If the front is propagated correctly, all the nodes in the gray area should be reached by the front. In order to do so, some nodes must be updated from a large distance. For example, due to the nature of the speed function, node \mathbf{x}_i in Fig. 12b can only be updated from locations in the transparent green area. The only node that can provide the update is the blue node itself. Consequently, this front can only be propagated correctly when nodes are updated from a much larger distance than only the direct neighbours. This is shown in Fig. 12c, where the arrows indicate the closest node from which a node can be updated, such that the direction of the update is above the overhang angle.

For structured meshes, on the other hand, the wedge-shaped speed function can be propagated efficiently. If the overhang angle does not align with the mesh, as in Fig. 12d, the situation is similar to unstructured meshes, and updates from a large distance might be required, as illustrated. However, when $\tan(\alpha_{oh}) = \Delta y/\Delta x$, where Δy and Δx are the element height and width, a node only has to be updated from its direct neighbours, as displayed in Fig. 12e. That is because if a node i can be updated from a far away node j , at least one of the direct neighbours below i can also be updated from node j (Fig. 7). For the speed function as given in (18), it is therefore sufficient to only consider the three nodes directly below the node of interest.

Altogether, if the filter is required to function on an unstructured mesh, the anisotropic speed function used in the layer-by-layer filter cannot be used. Therefore, a different speed function is chosen that instead of having

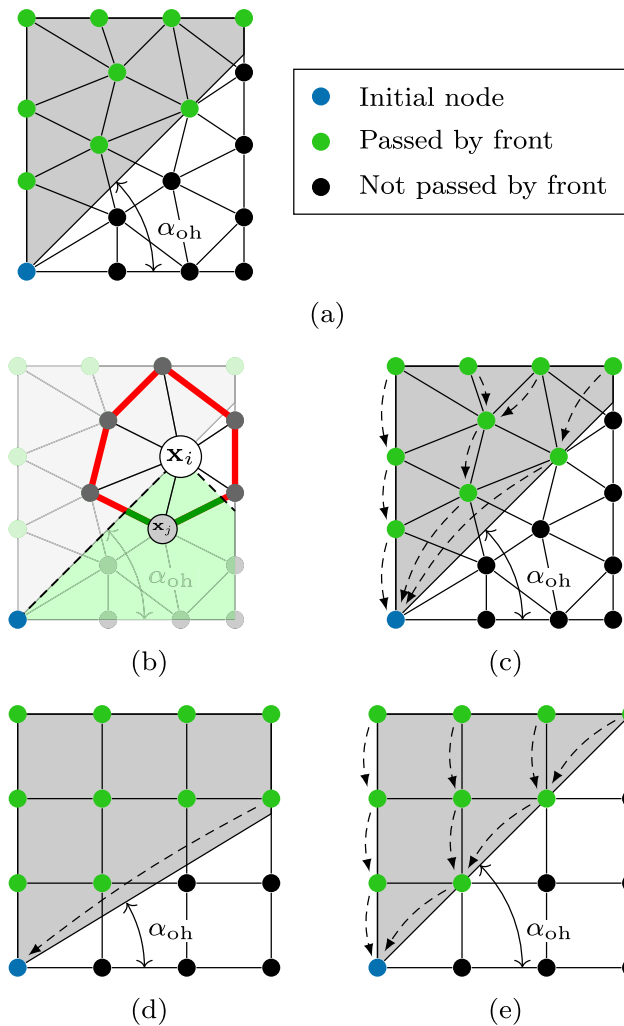
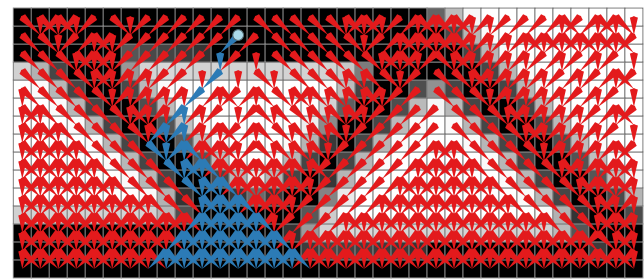


Fig. 12 Front propagation from a single node (blue) with a speed function as displayed in Fig. 6 for unstructured (a, b, c) and structured meshes (d, e). Nodes in the gray domain are reachable by the front, while black nodes are not

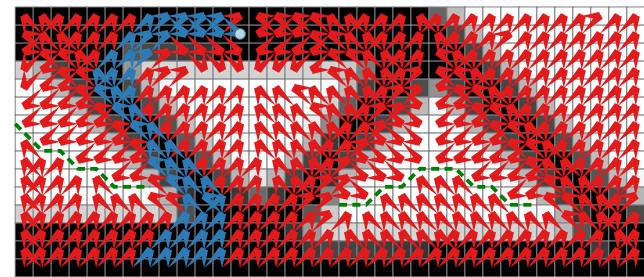
a zero propagation speed in the direction that are not printable, has a non-zero propagation speed in those directions. This requires a different processing of the resulting arrival times of the front propagation, which is discussed in Section 4.2.

4.1.1 Effect of the speed function on sensitivities

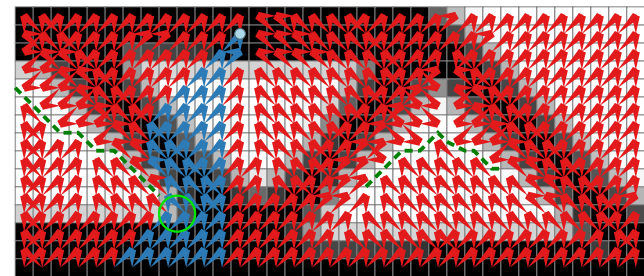
The speed function influences the direction in which the front propagates, which determines the order in which the nodes meet with the propagating front. This is important for the resulting sensitivities, as the sensitivity information spreads through the domain in exactly the opposite order as in which the front is propagated. This is shown in Fig. 13, where the layer-by-layer filter and the continuous filter have been applied to the same geometry shown in black-white. For every element, it is displayed from which other elements



(a) Layer-by-layer filter.



(b) Continuous filter with low v_{void} .



(c) Continuous filter with high v_{void} .

Fig. 13 The arrows pointing from each element indicate which elements are used to calculate its arrival time. This indicates to which elements sensitivity information is passed. For one element, highlighted by a blue circle, all the dependencies of elements that are involved in the calculation of its arrival time are coloured blue. **a** Layer-by-layer filter. **b** Continuous filter with low v_{void} . **c** Continuous filter with high v_{void}

its arrival time has been calculated, indicated by the arrows. For one element, indicated with a light-blue circle, all the nodes that could have contributed to this node arrival time are highlighted with blue arrows.

For the layer-by-layer filter (Fig. 13a), each element can only be updated from the three elements of the underlying layer (1). Since a smooth maximum is used to determine from which of the three lower elements it is updated, it can depend on up to three elements. This is usually the case when all the three underlying elements have the same printable density, for example for all the elements of the second row. On the other hand, in the implementation of the continuous filter as in van de Ven et al. (2018), no smooth minimum operator is used to select from which elements the arrival time is calculated (17). Therefore, each element

depends strictly on two other elements: the two elements that define the line segment from which the element was updated (16), which can be seen in Fig. 13b. When multiple update directions result in a similar arrival time, a single direction is chosen. This can be seen in, for example, the elements on the second row. Most of them are updated from the elements below-left and below. An update from the elements below-right and below would have resulted in the same arrival time, but this is neglected due to the strict minimum operator. Furthermore, contrary to the layer-by-layer filter, the continuous filter allows propagation in all directions. Therefore, one can find elements that have been updated from elements in the same or even higher layers, as is the case for the element indicated with the light-blue circle.

As the sensitivities are propagated in opposite direction of the front propagation, these differences have a major effect on the sensitivities, albeit the resulting arrival times are similar. In Fig. 13, all the elements that have a blue arrow are in some way involved in the calculation of the arrival time of the element indicated with the light-blue circle. In other words, in order to increase the printable density of the indicated element, the density of the blue-arrowed elements should be increased. For the layer-by-layer filter, this results in the creation of a support column through the void region in the middle. For the continuous filter, the column to left of the indicated element will be expanded towards this element. In this case, the layer-by-layer filter is more likely to form support columns, while the continuous filter will expand or shift existing column for support.

For the continuous filter, the speed with which the front propagates through void can be increased, as can be seen in Fig. 13c. The higher void speed shifts the region for which it is faster to travel directly to the base upwards (i.e. where elements are updated from below, as in the layer-by-layer filter), as indicated by the dashed green line in Fig. 13c and b. This will result in a more similar behaviour to the layer-by-layer filter. Finally, a higher void speed also allows the front to “cut corners”, as can be seen by the corner indicated with the green circle, which can be beneficial for the sensitivities, as these corner elements are usually the only elements whose density can be further increased, as can be seen in Fig. 13b for the blue-arrowed elements.

4.2 Differences in speed-density relations

One of the most attractive features of the layer-by-layer filter as compared to the continuous filter, is that an element can be supported by an underlying element with a similar or higher density. For example, when the layer-by-layer filter is applied to a uniform density field of $\rho = 0.5$, the resulting printable density field is identical: $\xi = 0.5$, as displayed in Fig. 14a and b. With the continuous filter, the front

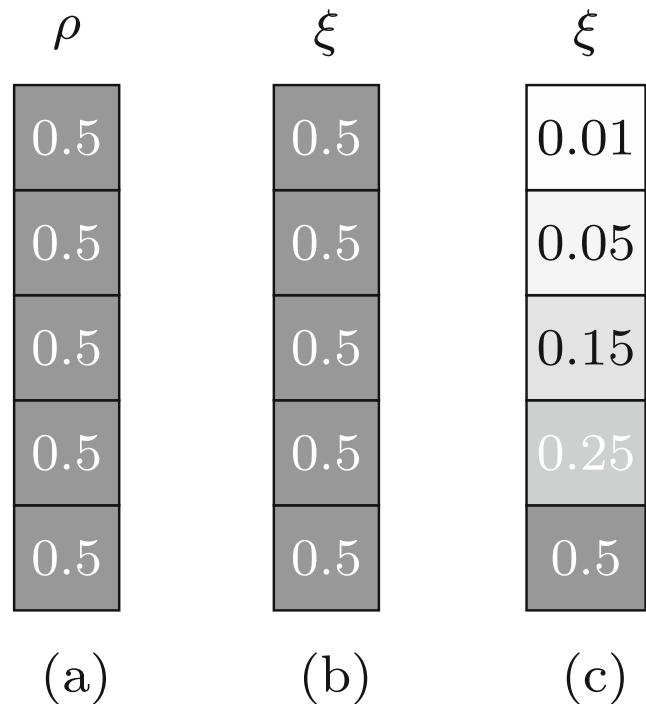


Fig. 14 Processing of a uniform intermediate density field (a), a typical initial condition of density-based topology optimization, by the layer-by-layer filter (b), and the continuous filter (c)

will travel at half speed through the domain, and the delay will increase with increasing height, resulting in a printable density field that decays with height, as can be seen in Fig. 14c. Since only material at the bottom of the domain is usually an unfavourable starting condition, it is proposed in van de Ven et al. (2018) to activate the continuous filter after 10 iterations.

The layer-by-layer filter achieves its behaviour by a smart choice of speed-density relation. As explained in Section 3.2, the propagation speed is only lowered when the current element has a lower density than the printable density of the supporting element. Otherwise, $g = 1$, maintaining a propagation speed equal to f_0 , the propagation speed used for the reference field T^{layer} . If the propagation speed is equal to f_0 , the delay does not increase, and the printable density is equal to that of the supporting element, resulting in the desired behaviour described by (2).

For the continuous filter, the propagation speed is simply scaled with the element’s density: when the element has full density, $g = 1$, and for intermediate densities $g < 1$. This results in a stricter penalization of intermediate densities: for every intermediate density element, $g < 1$ and the delay increases.

Certainly, the layer-by-layer speed-density relation is preferable for better convergence. In Section 5 it is investigated if the speed function used in the layer-by-layer

filter can be transferred to the continuous filter to improve its performance.

4.3 Smooth min/max approximations

The final difference between both methods that is compared in this work, is the implementation of the minimum/maximum operators of both methods to select which underlying element is the supporting element. For the layer-by-layer filter, this is the element with the highest printable density (1), and for the continuous filter the element with the lowest arrival time (17).

In Langelaar (2017), the discrete maximum operator is implemented as a penalized p-norm, to provide a smooth differentiable function. This enables the calculation of correct sensitivities when the arguments of the maximum operator are close to each other. In van de Ven et al. (2018), the minimum operator that selects the lowest arrival time is implemented as a discrete operator: only the lowest arrival time is used for the front propagation, and in the sensitivity analysis the adjoint is propagated only to the elements that provided the lowest arrival time. This leads to incorrect sensitivity information when the arrival time calculated from different nodes is equal, which is especially visible in the first iteration, where one starts with a uniform density field.

In order to investigate the effect of the discrete operator on the sensitivities, consider the volume constraint

$$f_1 = \sum_{i=1}^N \xi_i V_{el} / V_{tot} - V_{frac}, \quad (31)$$

where V_{el} and V_{tot} are the element and domain volume, respectively, and N is the total number of elements. For the volume constraint on a structured mesh, the derivatives to the printable densities are equal for every element:

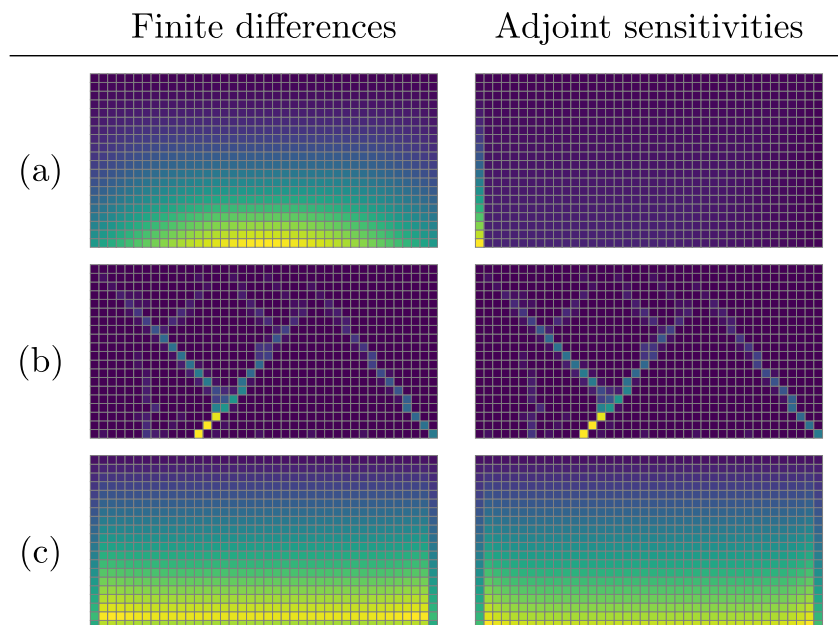
$$\frac{df_1}{d\xi_i} = V_{el} / V_{tot}, \quad \forall i. \quad (32)$$

The sensitivities towards the densities $\frac{df_1}{d\rho_i}$ for a uniform density field are displayed in Fig. 15a. The finite differences show the expected result, where densities towards the bottom have a higher sensitivity value, since more printable densities (and thus a large volume) are influenced by the lower elements than by the higher elements, as the front propagates bottom to top.

Unfortunately, the adjoint sensitivity analysis is not able to capture the sensitivity behaviour accurately, as can be seen in Fig. 15a. Although for every element the three underlying elements provide equal arrival times in the uniform density field, the sensitivity information is propagated to only one of the underlying elements because of the discrete minimum operator. In this case, the sensitivities are transferred to the lower left element because it has the lowest index. This results in an aggregation of sensitivities in the leftmost column, as displayed in Fig. 15a.

Fortunately, this is a problem that only occurs for a perfectly uniform density field. For example, when a tiny random perturbation with a magnitude of 1×10^{-5} is added to the density field, the adjoint sensitivities and finite differences are virtually identical. This can be seen in Fig. 15b, where instead of a uniform density field, a density field resulting from an MBB beam optimization is

Fig. 15 Comparison of finite differences and adjoint sensitivities for **a** the continuous filter applied to a uniform density field, **b** the continuous filter applied to a typical MBB beam result, and **c** the smoothed continuous filter applied to a uniform density field



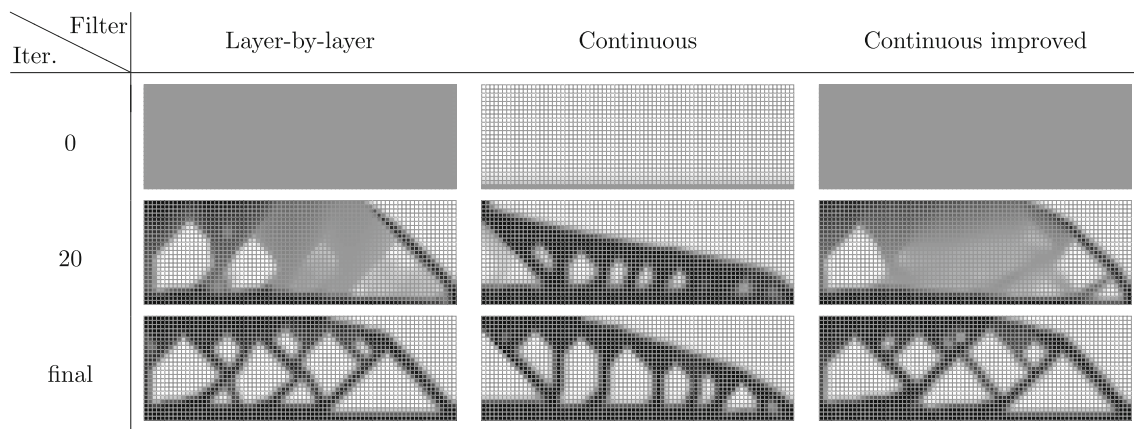


Fig. 16 Initial, intermediate and final designs of the different overhang filters for the MBB beam optimization problem

used, similar to the density field shown in Fig. 10a. As can be seen, there is a good correspondence between the finite differences and adjoint sensitivities, with a maximum relative difference of less than 0.1%.

It is however possible to formulate the front propagation with a smooth minimum operator. The implementation is mainly an exercise in bookkeeping as, contrary to the layer-by-layer filter, it is not known a priori from how many elements an element is updated. This is determined during the front propagation, based on the order in which the elements are updated. Therefore, one needs to keep track of the number of updates an element receives, from which elements these updates are done, the resulting arrival times of these updates, and the corresponding sensitivity information. The resulting adjoint sensitivities are much closer to the finite differences for a uniform field, as shown in Fig. 15c. As the smoothed minimum operator is tedious to implement and did not result in a significant improvement in convergence, it was not implemented in van de Ven et al. (2018).

5 Improved continuous overhang filter

In the previous section, a detailed comparison between the layer-by-layer filter and the continuous filter was performed, enabled by the front propagation formulation of the layer-by-layer filter. In this section, an improved version of the continuous front propagation filter is proposed, by incorporating elements of the front propagation formulation of the layer-by-layer filter.

In Section 4.1 it was shown that the wedge-shaped speed function used by the layer-by-layer filter can only be used on a structured grid, and is therefore not eligible for use in the continuous filter. However, the speed-density relation g (30) can be applied to the rectangular speed function of the continuous filter. For this, only one alteration is

required. For the layer-by-layer filter, the distance between two elements projected on the build direction is constant and appears as Δy in the speed-density relation (30). For the continuous filter, an element i is update from a location \mathbf{x}_y on a line segment between two other elements j and k , as displayed in Fig. 5. Therefore, the speed-density relation is formulated as:

$$g = \frac{|\mathbf{b} \cdot (\mathbf{x}_i - \mathbf{x}_y)|}{f_0 (h^{-1}(\min(\xi_y, \rho_i)) - h^{-1}(\xi_y)) + |\mathbf{b} \cdot (\mathbf{x}_i - \mathbf{x}_y)|}, \tag{33}$$

with

$$\xi_y = h(T_y - T_y^{\text{layer}}). \tag{34}$$

In principle, this speed-density relation can be interpreted similar to the layer-by-layer version, as described in Section 3.2. Compared to the old speed-density relation, this speed function is not strictly anisotropic. When $\xi_y \leq \rho_i$ on the whole line segment $\mathbf{x}_j\mathbf{x}_k$, the speed-density relation reduces to 1, and is isotropic (except for the edge case when $|\mathbf{b} \cdot (\mathbf{x}_i - \mathbf{x}_y)| = 0$, in which case the speed function should be defined as $g = 1$). However, when $\xi_y > \rho_i$, the speed-density relation tries to increase the delay such that $\xi_i = \rho_i$, which makes it dependent on \mathbf{x}_y and thus

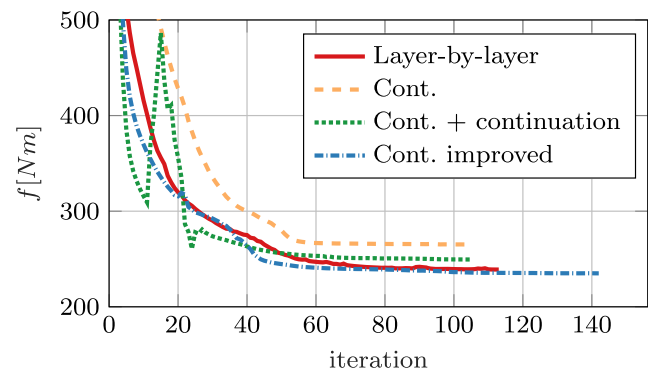


Fig. 17 Convergence behaviour of the different overhang filters

anisotropic. Therefore, one cannot use the same update algorithm as proposed in van de Ven et al. (2018). For this study, the minimization problem that has to be solved to find the minimal arrival time over the line segment $\mathbf{x}_j\mathbf{x}_k$ (16) is solved with brute force by probing 10 locations along the line segment and accepting the minimum value. For practical applications, a similar solution as in van de Ven et al. (2018) should be constructed to speed up the front propagation.

Another effect of using the speed-density relation as given in (33), is that the choice of the delay-density relation h (34), as long as it is monotonically decreasing, is not relevant: the speed-density relation g will reduce the equation for printable densities to a minimum formulation (24), independent of h . Therefore, the parameter k that had to be set for the continuous overhang filter in van de Ven et al. (2018) (6) is no longer required. This simplifies the formulation and makes it more robust, as a user-defined parameter is eliminated.

5.1 Numerical results

First, the layer-by-layer filter and the conventional and improved continuous filter are compared on a 2D problem,

after which the improved continuous filter is demonstrated in 3D.

5.1.1 2D case—comparison

The different overhang filters are compared on the compliance optimization of the MBB beam with 3:1 aspect ratio, with a volume constraint set to 50%. A density filter (Bruns and Tortorelli 2001; Sigmund and Petersson 1998) with a filter radius $r = 2\Delta x$ is applied, intermediate densities are penalized with SIMP penalization (Bendsøe 1989; Bendsøe and Sigmund 1999) of $p = 3$, and the Method of Moving Asymptotes (Svanberg 1987) is used as optimization algorithm. The optimizations are terminated when the maximum change of printable densities is smaller than 0.01.

The printable densities at the iteration 0, iteration 20, and final designs for the three methods are displayed in Fig. 16. The initial condition is a uniform density field with $\rho = 0.5$. It can be seen that both the layer-by-layer filter and the improved continuous filter allow the stacking of intermediate densities, and thus have a resulting initial field of $\xi = 0.5$. The conventional implementation of the continuous filter does not allow this and therefore only

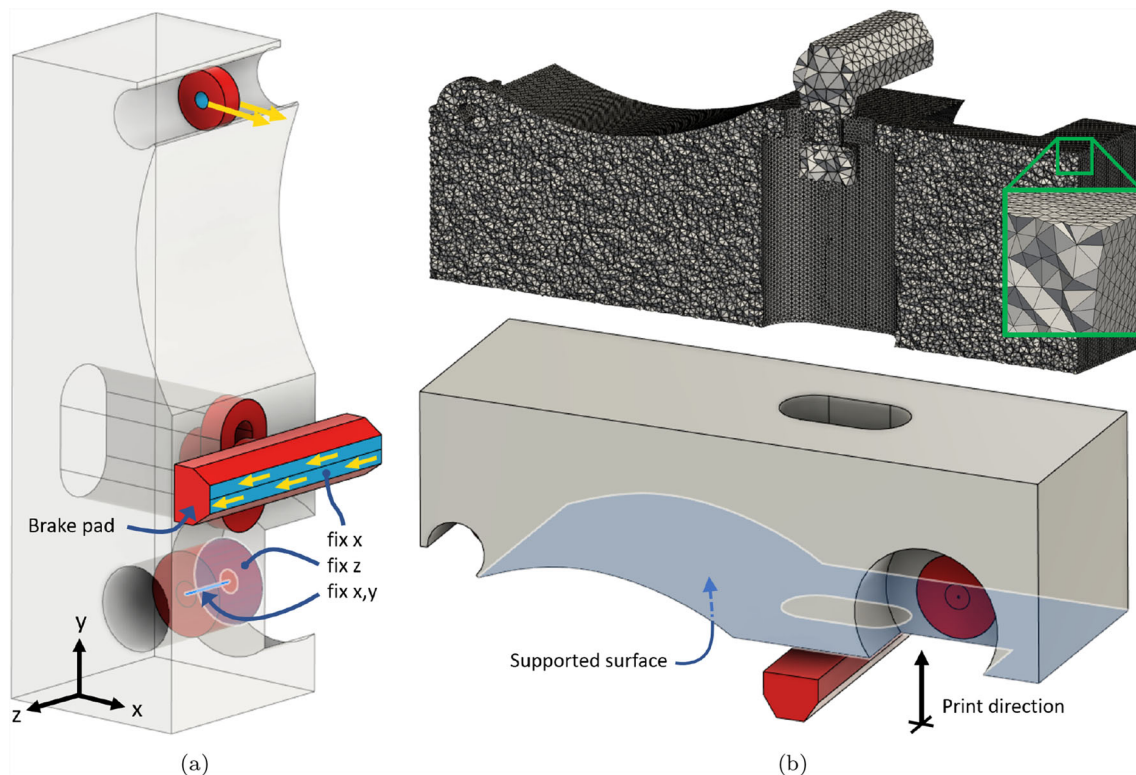


Fig. 18 The bicycle pull brake case. The fixed regions are shown in red, while the design domain is shown in gray. Applied forces are indicated by yellow arrows and fixed regions are indicated in (a). The

domain is meshed with tetrahedral elements, of which a clipped section is shown in (b) and is to be printed in the orientation as shown in (b)

the bottom layers remain. This is an unfavourable starting condition, as the design now has to grow from the bottom, as can be seen in the following snapshots. Eventually the optimization terminates in an suboptimal local minimum, as can be seen in Fig. 17. This behaviour is very similar to the overhang filter presented in Gaynor and Guest (2016).

The layer-by-layer filter and the improved continuous filter do not suffer from the poor initial condition, and can straight away put material at the top of the domain, as can be seen in Fig. 16. Both methods finalize in similar designs with similar objective values, as displayed in Fig. 17. Finally, the convergence behaviour of the conventional continuous filter with gradual activation after 10 iterations, as proposed in van de Ven et al. (2018), is also plotted in Fig. 17. As the filter is activated, the objective spikes, but converges to a lower objective than without continuation. However, the final objective is still higher than the layer-by-layer filter and the improved continuous filter.

5.1.2 3D case—bicycle pull brake

To fully demonstrate the capabilities of the improved continuous filter, it is demonstrated on a compliance case with a more challenging design domain, such as one might encounter in real-life problems. For the 3D case a linear pull brake for a bicycle is optimized for stiffness against a 10% volume constraint. The case will be optimized without an overhang constraint and with a 45° overhang constraint. The 3D optimization is performed with an optimization code based on the Portable, Extensible Toolkit for Scientific Computing (PETSc) (Balay et al. 2017; Balay et al. 1997; Karypis and Kumar 1998), and visualized with ParaView (Ahrens et al. 2005). Isosurfaces of the printable density field at $\xi = 0.5$ are depicted for visualization. Finally, the PETSc-based MMA implementation presented in Aage and Lazarov (2013) and Aage et al. (2015) has been used as optimization algorithm.

The design domain for the pull brake is shown in Fig. 18. The red regions are fixed regions, while the gray region is

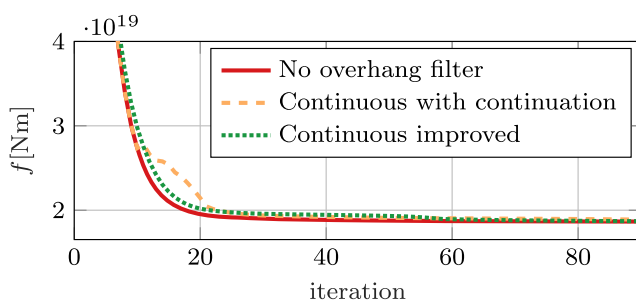


Fig. 19 Convergence behaviour for the first 90 iterations of the 3D case

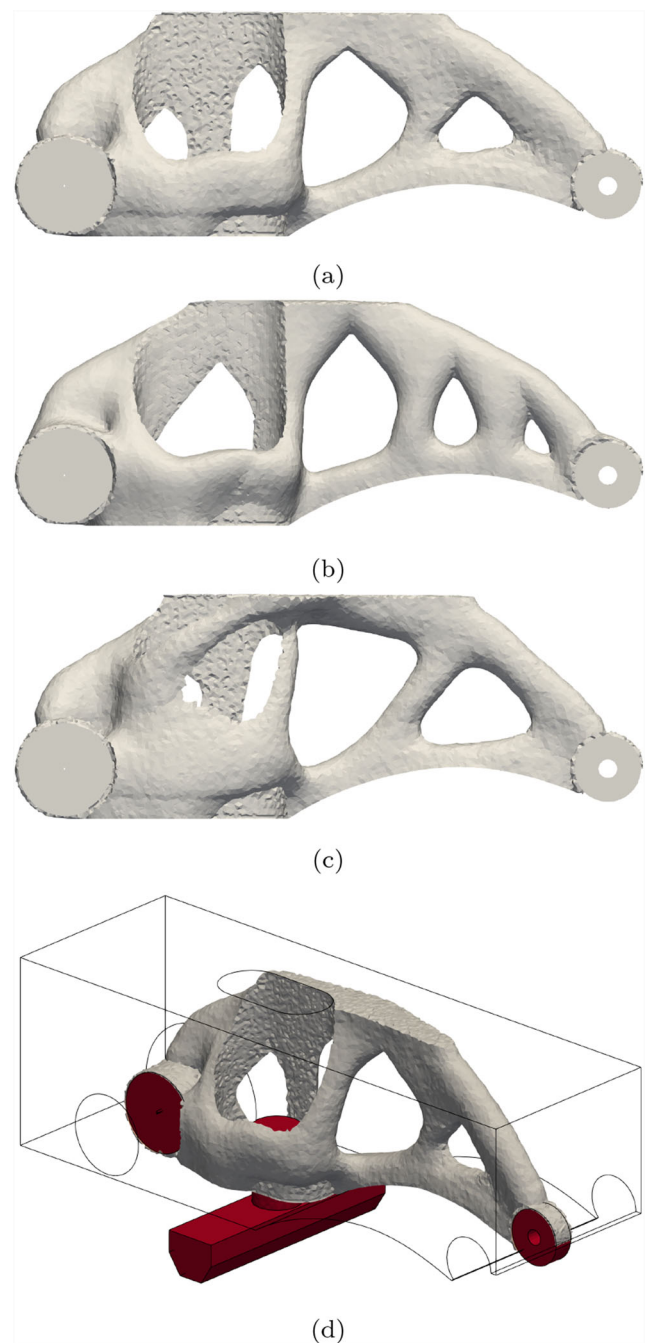


Fig. 20 Optimized geometries of the bicycle pull brake with: improved continuous filter (a), the old continuous filter with continuation (b), and without overhang filter (c). The geometry of the improved continuous filter is also shown with outlines of the original design domain, where the fixed density regions are shown in red (d)

the design domain. The pull brake is mounted to the bicycle with a bolt at the lowest fixed region in Fig. 18a. This is simulated by fixing the centre line of the bolted region in x- and y-direction, allowing rotation around the z-axis, and by fixing the back side in z-direction as indicated. When the brake is applied, a clamping force of $F_{\text{clamp}} = 200\text{N}$ is

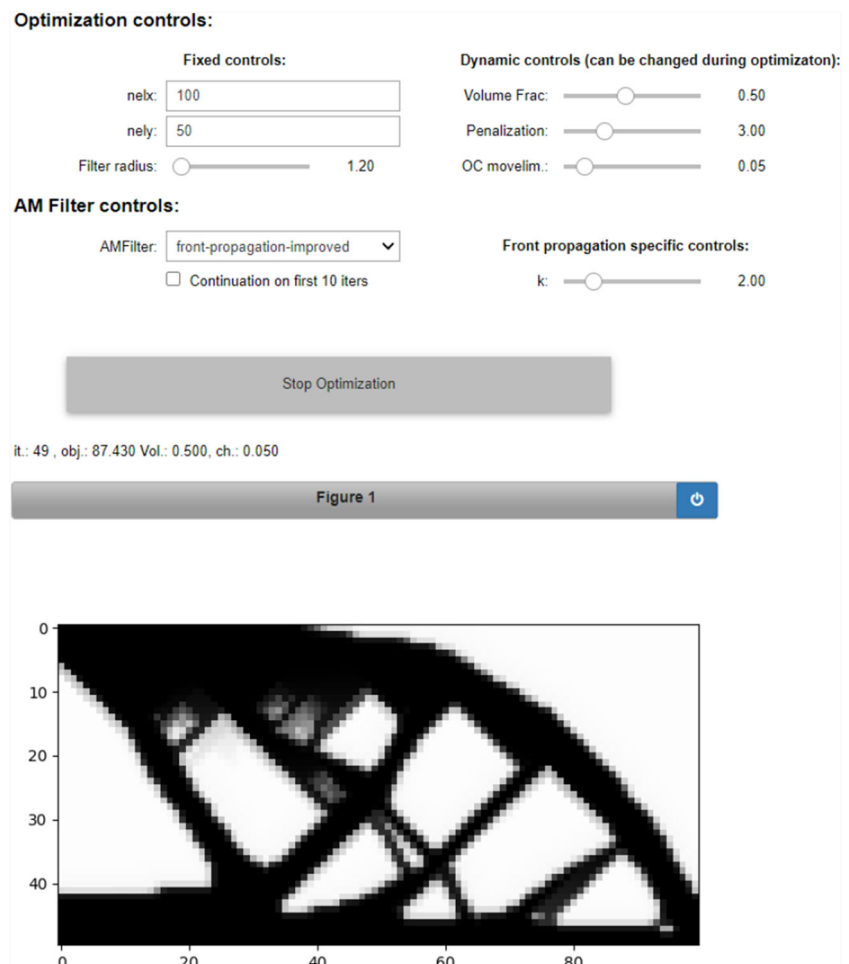
exerted on the top fixed region, indicated by the two yellow arrows in x-direction. This will push the brake pad against the wheel. Therefore, the brake pad is fixed in x-direction, and the brake force exerted by the tire on the brake pad is applied in z-direction, indicated by the yellow arrows in z-direction. The force on the brake pad is proportional to the clamping force: due to the geometry of the lever mechanism, the clamping force is amplified by a factor 3.7, which gives $F_{\text{brake}} = 3.7\mu F_{\text{clamp}}$, where μ is the friction coefficient assumed to be 0.5. The mechanism is to be printed (without the brake pad) in the orientation shown in Fig. 18b. A small amount of support will be required to manufacture the curved section of the supported surface, but since metal components are generally printed on a small layer of supports, this is not introducing a large amount of additional support material.

The irregular design domain of the pull brake shows the advantage of being able to restrict overhanging features on unstructured meshes. Standard meshing algorithms can be used to create a tetrahedral mesh, and fixed regions such as the brake pad can be meshed with a lower resolution to reduce the total number of nodes, as shown in Fig. 18b. In

this case, the domain is meshed using Gmsh (Geuzaine and Remacle 2009), with 2.7×10^5 nodes and 1.5×10^6 elements.

The optimized geometries are shown in Fig. 20. It can be seen that the geometries optimized with overhang filter can be printed in the given orientation without support material, while the geometry without overhang constraint has some overhanging members. The optimizations ran for 300 iterations; however, after 90 iterations, there is little change in the design and the objective values of all three optimizations are within 1% of their final values. The convergence behaviour for the first 90 iterations is shown in Fig. 19. The convergence curve is smooth for the improved continuous filter, as there is no need for continuation. For the continuous filter which requires continuation there is some non-smooth behaviour between iterations 10 and 20 where the continuation is applied, but the impact on the final result is negligible. It can however happen that the continuation is too aggressive, and the design has to be built up from the bottom of the domain, as was the case in Fig. 16 for the continuous filter without continuation. Therefore, the improved filter is a more stable version of the continuous filter because it does not require continuation. The final

Fig. 21 Snapshot of the Jupyter notebook provided in the [Supplementary Material](#)



objective values for the optimizations with the continuous improved filter and continuous filter with continuation are 0.6% and 1.0% higher than without overhang filter, respectively. It is however difficult to compare these values as the overhang filter slightly alters the length scale introduced by the density filter (Fig. 20).

6 Conclusion

In this paper, a detailed comparison is made between the layer-by-layer overhang filter as proposed in Langelaar (2017), and the continuous overhang filter as proposed in van de Ven et al. (2018). It is demonstrated that the layer-by-layer filter, originally formulated in a discrete setting, can be formulated in a continuous setting using front propagation. Similar to the continuous filter, the printable densities are based on a delay between two arrival time fields. As such, the layer-by-layer filter can be formulated using front propagation by only changing the speed function, and the speed-density relation.

The front propagation formulation of the layer-by-layer filter allows a component-level comparison between the two methods. It is shown that the speed function used by the layer-by-layer function cannot be applied to unstructured meshed due to the large anisotropy of the speed function. However, the speed-density relation of the layer-by-layer filter has the advantageous characteristic that it allows the support of intermediate density elements by elements of the same or higher density, improving convergence in the early stage of the optimization. Additionally, the different implementations of the minimum/maximum operators are investigated. As the continuous filter does not use a smooth minimum operator, the resulting sensitivities for a uniform density field are incorrect. It is possible to use smoothed operators, but in practice the difference is small, as the sensitivities are only inaccurate for a uniform density field: a minutely perturbed density field results in correct sensitivities.

Finally, the speed-density relation used by the layer-by-layer filter is transferred to the continuous overhang filter, resulting in an overhang filter with the advantageous characteristics of the layer-by-layer filter, but directly applicable to unstructured meshes.

A Matlab implementation of the continuous filter is provided with this work, to aide with the implementation of a front propagation-based overhang filter in any existing density-based topology optimization.

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.1007/s00158-021-02887-2>.

Acknowledgements The authors thank Krister Svanberg for use of the MMA optimizer.

Funding This research was carried out under project number S12.7.14549a in the framework of the Metals for Additive Manufacturing Partnership Program of the Materials innovation institute M2i (www.m2i.nl) and the Netherlands Organization for Scientific Research (www.nwo.nl).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Replication of results An implementation of the front propagation-based overhang filter presented in this work is provided in the form of an interactive Jupyter notebook, shown in Fig. 21. It is based on the Python implementation of the 88-line topology optimization code (Andreassen et al. 2011). It contains the front propagation-based overhang filter as presented in van de Ven et al. (2018), the improved variant presented in this work, and a Python adaption of the layer-by-layer filter provided in Langelaar (2017). In order to run the notebook, open a Jupyter notebook (www.jupyter.org), inside the notebook navigate to the [Supplementary Material](#) and run `TopOptAMapplet.ipynb`. Then follow the instructions inside the notebook. The reader is encouraged to play with the various overhang filters and parameters to gain an understanding of their effect on the optimization.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Aage N, Lazarov BS (2013) Parallel framework for topology optimization using the method of moving asymptotes. *Struct Multidiscip Optim* 47(4):493–505
- Aage N, Andreassen E, Lazarov BS (2015) Topology optimization using PETSc: An easy-to-use, fully parallel, open source topology optimization framework. *Struct Multidiscip Optim* 51(3):565–572
- Adam GA, Zimmer D (2014) Design for additive manufacturing—element transitions and aggregated structures. *CIRP J Manuf Sci Technol* 7(1):20–28
- Ahrens J, Geveci B, Law C (2005) Paraview: an end-user tool for large-data visualization. In: Hansen CD, Johnson CR (eds) *Visualization handbook*. Butterworth-Heinemann, Burlington, pp 717–731
- Allaire G, Dapogny C, Estevez R, Faure A, Michailidis G (2017) Structural optimization under overhang constraints imposed by additive manufacturing technologies. *J Comput Phys* 351:295–328
- Amir O, Mass Y (2018) Topology optimization for staged construction. *Struct Multidiscip Optim* 57(4):1679–1694
- Andreassen E, Clausen A, Schevenels M, Lazarov BS, Sigmund O (2011) Efficient topology optimization in MATLAB using 88 lines of code. *Struct Multidiscip Optim* 43(1):1–16

- Balay S, Gropp WD, McInnes LC, Smith BF (1997) Efficient management of parallelism in object oriented numerical software libraries. In: Arge E, Bruaset AM, Langtangen HP (eds) *Modern Software Tools in Scientific Computing*. Birkhäuser Press, pp 163–202
- Balay S, Abhyankar S, Adams MF, Brown J, Brune P, Buschelman K, Dalcin L, Eijkhout V, Gropp WD, Kaushik D, Knepley MG, McInnes LC, Rupp K, Smith BF, Zampini S, Zhang H, Zhang H (2017) *PETSc users manual*. Tech. Rep. ANL-95/11 - Revision 3.8, Argonne National Laboratory
- Bendsøe MP (1989) Optimal shape design as a material distribution problem. *Struct Optim* 1(4):193–202
- Bendsøe MP, Sigmund O (1999) Material interpolation schemes in topology optimization. *Archiv Appl Mech (Ingenieur Archiv)* 69(9-10):635–654
- Bendsøe MP, Sigmund O (2004) *Topology optimization*. Springer Berlin Heidelberg, Berlin, Heidelberg
- Bruns TE, Tortorelli DA (2001) Topology optimization of non-linear elastic structures and compliant mechanisms. *Comput Methods Appl Mech Eng* 190(26-27):3443–3459
- Cloots M, Spierings A, Wegener K (2013) Thermomechanisches Multilayer-Modell zur Simulation von Eigenspannungen in SLM-Proben p 11
- Gaynor AT, Guest JK (2016) Topology optimization considering overhang constraints: Eliminating sacrificial support material in additive manufacturing through design. *Struct Multidiscip Optim* 54(5):1157–1172
- Geuzaine C, Remacle JF (2009) Gmsh: a 3-D finite element mesh generator with built-in pre- and post-processing facilities. *Int J Numer Meth Eng* 79(11):1309–1331
- Guo X, Zhou J, Zhang W, Du Z, Liu C, Liu Y (2017) Self-supporting structure design in additive manufacturing through explicit topology optimization. *Comput Methods Appl Mech Eng* 323:27–63
- Hoffarth M, Gerzen N, Pedersen C (2017) ALM overhang constraint in topology optimization for industrial applications. In: *Proceedings of the 12th World Congress on Structural and Multidisciplinary Optimisation*. Braunschweig, Germany, p 12
- Jiang J, Stringer J, Xu X, Zhong RY (2018) Investigation of printable threshold overhang angle in extrusion-based additive manufacturing for reducing support waste. *Int J Comput Integr Manuf* 31(10):961–969
- Karypis G, Kumar V (1998) A parallel algorithm for multilevel graph partitioning and sparse matrix ordering. *J Parallel Distrib Comput* 48:71–85
- Kranz J, Herzog D, Emmelmann C (2015) Design guidelines for laser additive manufacturing of lightweight structures in TiAl6V4. *J Laser Appl* 27(S1):S14,001
- Langelaar M (2016) Topology optimization of 3D self-supporting structures for additive manufacturing. *Addit Manuf* 12:60–70
- Langelaar M (2017) An additive manufacturing filter for topology optimization of print-ready designs. *Struct Multidiscip Optim* 55(3):871–883
- Langelaar M (2018) Combined optimization of part topology, support structure layout and build orientation for additive manufacturing. *Struct Multidiscip Optim* 57(5):1985–2004
- Liu J, Gaynor AT, Chen S, Kang Z, Suresh K, Takezawa A, Li L, Kato J, Tang J, Wang CCL, Cheng L, Liang X, To AC (2018) Current and future trends in topology optimization for additive manufacturing. *Struct Multidiscip Optim* 57(6):2457–2483
- Mercelis P, Kruth JP (2006) Residual stresses in selective laser sintering and selective laser melting. *Rapid Prototyp J* 12(5):254–265
- Qian X (2017) Undercut and overhang angle control in topology optimization: a density gradient based integral approach. *Int J Numer Methods Eng* 111(3):247–272
- Ranjan R, Ayas C, Langelaar M, van Keulen A (2018) Towards design for precision additive manufacturing: a simplified approach for detecting heat accumulation. In: *Proceedings of the ASPE and EUSPEN summer topical meeting*
- Sethian JA, Vladimirsky A (2003) Ordered upwind methods for static Hamilton–Jacobi equations: Theory and algorithms. *SIAM J Numer Anal* 41(1):325–363
- Sigmund O, Petersson J (1998) Numerical instabilities in topology optimization: A survey on procedures dealing with checkerboards, mesh-dependencies and local minima. *Struct Optim* 16(1):68–75
- Sih SS, Barlow JW (2004) The prediction of the emissivity and thermal conductivity of powder beds. *Part Sci Technol* 22(4):427–440
- Svanberg K (1987) The method of moving asymptotes—a new method for structural optimization. *Int J Numer Methods Eng* 24(2):359–373
- Thomas D (2009) *The Development of Design Rules for Selective Laser Melting* PhD thesis. University of Wales, Cardiff
- van de Ven E, Maas R, Ayas C, Langelaar M, van Keulen F (2018) Continuous front propagation-based overhang control for topology optimization with additive manufacturing. *Struct Multidiscip Optim* 57(5):2075–2091
- van de Ven E, Ayas C, Langelaar M, Maas R, van Keulen F (2018) A pde-based approach to constrain the minimum overhang angle in topology optimization for additive manufacturing. In: Schumacher A, Vietor T, Fiebig S, Bletzinger KU, Maute K (eds) *Advances in structural and multidisciplinary optimization*. Springer International Publishing, Cham, pp 1185–1199
- Wang D, Yang Y, Yi Z, Su X (2013) Research on the fabricating quality optimization of the overhanging surface in SLM process. *Int J Adv Manuf Technol* 65(9-12):1471–1484
- Zhang K, Cheng G, Xu L (2019) Topology optimization considering overhang constraint in additive manufacturing. *Comput Struct* 212:86–100

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.