HAL
open science

Ofast3D - Étude de faisabilité

Damien Hardy

To cite this version:

Damien Hardy. Ofast3D - Étude de faisabilité. [Rapport Technique] RT-0511, Inria Rennes - Bretagne Atlantique; IRISA. 2020, pp.18. hal-03093905

HAL Id: hal-03093905

https://inria.hal.science/hal-03093905v1

Submitted on 4 Jan 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Ofast3D – Étude de faisabilité

Damien Hardy

TECHNICAL REPORT
N° 0511
December 2020
Project-Team PACAP

ISSN 0249-0803 ISRN INRIA/RT--0511--FR+ENG

Ofast3D – Étude de faisabilité

Damien Hardy

Équipe-Projet PACAP

Rapport technique n° 0511 — December 2020 — 18 pages

Résumé : Ce rapport technique présente les résultats de l'étude de faisabilité du projet Ofast3D menée de mai à juillet 2020 dans le cadre de la lutte contre la Covid-19. L'objectif de Ofast3D est d'augmenter la capacité de production des acteurs professionnels utilisant l'impression 3D par dépôt de filament, sans avoir à modifier les infrastructures de production existantes. Cette étude présente plusieurs pistes d'optimisation permettant d'atteindre cet objectif. Une première optimisation est mise en œuvre pour réduire les déplacements à vide. Des tests en conditions réelles montrent de bons potentiels pour réduire les temps d'impression sans affecter la qualité sur des modèles 3D simples.

Mots-clés : impression 3D, dépôt de filament, temps d'impression

RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

Ofast3D – Feasibility Study

Abstract: This technical report presents the results of the feasibility study of the Ofast3D project conducted from May to July 2020 as part of the fight against Covid-19. The goal of Ofast3D is to increase the production capacity of professional players using fused deposition modeling 3D printing, without requiring any modification of existing production infrastructures. This study presents several optimization opportunities to achieve this objective. A first optimization is implemented to reduce airtime movements. Experiments in real conditions show a good potential for reducing printing times without affecting the quality on simple 3D models.

Key-words: 3D printing, fused deposition, printing time

Ce document synthétise les actions menées lors de l'étude de faisabilité du projet Ofast3D de mai à juillet 2020. Il replace le contexte de cette étude (Sec. 1). Une vue d'ensemble de l'impression 3D est présentée (Sec. 2) et situe le positionnement de cette étude (Sec. 3). Après un bref comparatif du comportement de différents trancheurs sur le cas d'une visière de protection (Sec. 4), les différentes réalisations logicielles et la faisabilité d'une première optimisation sont détaillées ainsi que des pistes pour d'autres optimisations futures (Sec. 5) avant de dresser un bilan (Sec. 6) et des perspectives (Sec. 7).

Cette étude a été effectuée avec Erven Rohou, et la participation de Guillermo Andrade-Barroso, Loïc Besnard, et Laurent Garnier ainsi qu'un partenaire professionnel.

1 Contexte de l'étude

Pendant le premier pic de l'épidémie de la Covid-19 que nous avons connu en France, il y a eu un besoin de produire en urgence des produits permettant de lutter efficacement contre ce virus. L'impression 3D a permis de répondre, en partie, à ce besoin. En effet, il y a eu une utilisation intensive de cette technologie par des particuliers, des fablabs, des laboratoires de recherche et des professionnels pour produire des solutions innovantes (visières de protection, divers adaptateurs en lien avec les respirateurs...) permettant ainsi d'apporter des solutions immédiates pour faire face à une pénurie de matériels spécialisés pour le corps médical.

Depuis, nous avons pu également observer l'émergence de nouvelles demandes (système d'ouverture de porte sans contact direct, distributeur de gel hydroalcoolique sans contact, porte-élastique...) liées principalement à la problématique du déconfinement pour prévenir et ainsi éviter une deuxième vague de contamination potentielle.

L'une des principales limites, une fois les solutions identifiées, est la capacité de production pour répondre aux demandes massives. L'objectif de Ofast3D est d'augmenter la capacité de production des professionnels utilisant l'impression 3D par dépôt de filament, sans avoir à modifier les infrastructures de production existantes. Elle consiste à optimiser le code interprété par les imprimantes 3D en prenant en compte la géométrie des modèles 3D ainsi que l'imprimante cible pour réduire le temps d'impression. Cette approche est complémentaire aux méthodes visant soit à améliorer les imprimantes soit à optimiser les modèles 3D eux-mêmes.

2 Impression 3D vue d'ensemble

Différents procédés existent concernant la fabrication additive également appelée impression 3D. Les trois procédés principaux sont le dépôt de filament, la stéréolithographie et le frittage laser sélectif. Chaque procédé a ses avantages mais a également des limites ainsi que des coûts et des usages différents. Cette étude se concentre uniquement sur le procédé dit à dépôt de filament.

Le procédé par dépôt de filament consiste à déposer du filament fondu par couches successives afin d'imprimer le modèle souhaité. Chaque couche est constituée d'une multitude de segments pouvant correspondre à des parois (interne ou externe), du remplissage ou des supports.

Pendant l'impression, le filament est poussé par un extrudeur vers un corps de chauffe (*hotend*) avant d'être déposé par une buse soit sur un plateau dans le cas de la première couche soit sur les couches inférieures. Différents types de modèles d'imprimantes 3D par dépôt de filament existent notamment : cartésienne, hypercube, core XY et delta. La principale différence réside dans les parties de l'imprimante qui sont en déplacement et comment ils sont déplacés suivant les axes XYZ lors de l'impression (plateau et hotend). L'extrudeur peut également être soit déporté (bowden) soit proche (direct-drive) du corps de chauffe. Cette étude est axée sur des modèles

G91 ; *relative pos*	**G90** ; *absolute pos*
M82 ; *E absolute*	**M83** ; *E relative*
G1 F3000 X100 E10	**G1** F3000 X100 Y0 E10
G1 Y100 E20	**G1** X100 Y100 E10
G1 X−100 E30	**G1** X0 Y100 E10
G1 Y−100 E40	**G1** X0 Y0 E10

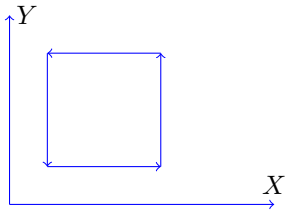


TABLE 1 – exemple de commande G-code réalisant le parcours d'un carré soit en relatif soit en absolu

cartésien et hypercube en direct-drive. Néanmoins, il n'y a pas a priori de raison apparente pour que cela ne soit pas applicable à l'ensemble de ces configurations.

Les modèles sont généralement modélisés par des logiciels de CAO/FAO. D'autres méthodes, comme l'utilisation d'un scanner 3D ou l'utilisation de scripts, peuvent également être utilisées. Le modèle est ensuite exporté dans un format de fichier comme STL qui décrit la surface externe du modèle par des triangles. Le fichier STL est alors transmis à un trancheur. Le trancheur est en charge de découper le modèle 3D en couches successives et en fonction des paramètres de l'imprimante cible et des paramètres sélectionnés par l'utilisateur de générer le fichier G-code pour l'imprimante 3D. Dans les étapes de génération du G-code par le trancheur, il y a notamment la définition de l'ordre du parcours d'impression ainsi que la quantité de matière à déposer sur ce parcours.

Le G-code est un langage de bas niveau décrivant des actions à réaliser par des machines CNC dont les imprimantes 3D. G-code est standardisé par le NIST sous la référence RS274. Le fichier G-code est un fichier texte décrivant l'ensemble des mouvements successifs par une liste d'instructions G-code, typiquement des centaines de milliers, pour imprimer le modèle 3D. Il est interprété par le *firmware* de l'imprimante et est constitué d'une suite de commandes concernant les déplacements suivants au moins quatre axes (X, Y, Z et E pour l'extrudeur) ainsi que des commandes de température, de configuration de l'imprimante et des commentaires.

Finalement, le coût d'impression peut se décomposer en trois parties :

— avant l'impression : de la définition du modèle jusqu'à la préparation de l'impression ;
— l'impression elle-même ;
— après l'impression : le retrait du modèle de l'imprimante ainsi que le post traitement comme le retrait des supports, le ponçage, un traitement chimique éventuel.

Pour de plus amples informations sur l'ensemble de ce processus [22] donne une très bonne vue d'ensemble.

3 Positionnement de l'étude

L'étude se concentre sur la toute fin de la partie avant impression. Nous supposons que les optimisations portant sur les imprimantes ou les modèles 3D à imprimer sont déjà réalisées. De plus, les paramètres du trancheur (environ 600 pour Cura version 4.4.1) sont bien choisis pour le modèle par l'utilisateur éventuellement assisté par d'autres outils comme [1]. Finalement, le placement de la pièce, le positionnement des supports et le remplissage sont également supposés bien définis en fonction du modèle et de l'usage ultérieur de celui-ci.

L'étude s'intéresse à la génération du G-code effectuée par le trancheur et vise à identifier des

opportunités d'optimisations pouvant s'effectuer dans le trancheur lui-même ou ultérieurement en optimisant le fichier G-code directement.

Les optimisations ont évidemment pour but de réduire le temps d'impression du modèle. Par contre elles ne doivent pas impacter la qualité ou de façon négligeable pour ne pas affecter le temps de post traitement.

Quelques travaux de recherche similaires portant sur les CNC en général [16] ou sur les imprimantes 3D à dépôt de filament en particulier [26, 21, 20, 25] ont déjà été réalisés soit dans le trancheur soit sur le G-code directement. Ce type de travaux est néanmoins absent des versions actuelles des trancheurs (cf. section 5) et il n'existe pas a priori de logiciel mature pour effectuer ce type d'optimisations.

4 Comportement des différents trancheurs sur le cas des visières de protection

Le point de départ de cette étude porte sur les déplacements à vide lors de l'impression de visières de protection en utilisant le G-code généré par le logiciel Ultimaker Cura 4.4.1. Afin de s'assurer que ce n'est pas seulement un cas spécifique à ce logiciel nous avons également utilisé d'autres trancheurs pour observer leur comportement vis-à-vis des déplacements à vide. L'annexe 1 donne une illustration des déplacements à vide observables pour un modèle de visière type V30 [2] pour les trancheurs Ultimaker Cura 4.4.1, IceSL 2.3.4, IdeaMaker 3.5.3.4250, PrusaSlicer 2.2.0 et Simplify3D 4.1.2. Ce comportement est clairement non spécifique à un trancheur en particulier.

L'une des raisons possibles provient du fait que les utilisateurs d'un trancheur souhaitent avoir très rapidement une représentation, une estimation du temps d'impression et de quantité de matière du modèle à imprimer. Dans le cas de professionnels, cela permet par exemple d'itérer sur les différents réglages ou de faire une estimation du tarif des prestations quasi instantanément.

D'un autre côté, la définition du parcours s'apparente à un problème de voyageur de commerce connu pour être NP-complet. Une solution rapide ne sera de fait pas toujours proche d'une solution efficace dans le sens du parcours. Dans Ultimaker Cura 4.4.1, l'algorithme mis en œuvre est de type glouton et sélectionne le plus proche voisin.

5 Réalisation

Dans cette section, nous présentons les différentes réalisations effectuées pendant cette étude de faisabilité. Différents modèles 3D au format STL en lien avec la lutte contre la Covid-19 utilisés dans cette étude sont d'abord présentés. Une description des étapes clés du processus de tranchage utilisé dans Ultimaker Cura ainsi que la réalisation d'un script d'automatisation sont ensuite détaillées. Nous présentons ensuite le prototype d'analyse de langage G-code ainsi que différentes analyses et une première optimisation. Pour conclure cette partie, nous proposons d'autres pistes d'optimisations identifiées lors de l'étude.

5.1 Modèles 3D en lien avec la lutte contre la Covid-19

Afin d'effectuer des tests, nous avons retenu les modèles présentés dans le tableau 2. Les modèles représentent principalement différents types de visières mais d'autres modèles simples [3, 4] ont également été pris en compte pour avoir une première estimation de l'impact de la forme des modèles.

Nom	Description
Newshield [5]	Visière de protection type croco
V17 [6]	Visière de protection avec clip version 17
V30 [2]	Visière de protection version 30
Support lunette [7]	Support des lunettes de protection pour chirurgiens
Porte élastique [3]	Système de maintien des élastiques de masques de protection
Crochet [4]	Crochet personnel ouvre porte

Table 2 – modèles STL créés pour lutter contre la Covid-19

5.2 Ultimaker Cura et CuraEngine

Le logiciel Ultimaker Cura [8] est constitué d'une interface graphique permettant à l'utilisateur de charger un modèle 3D, de paramétrer l'imprimante 3D ainsi que les paramètres d'impression et de réaliser le tranchage. Le tranchage en lui-même est réalisé par le logiciel CuraEngine [9]. Ultimaker Cura peut ainsi être vu comme un frontend gérant un peu plus de 600 paramètres transmis à CuraEngine pour effectuer le tranchage. [10] décrit les principaux paramètres de tranchage.

L'algorithme de tranchage de CuraEngine comporte cinq étapes principales :

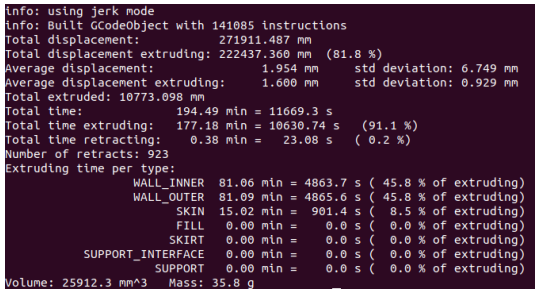
— le tranchage en couches (slicing) ;
— la génération des zones par couche ;
— le remplissage des zones ;
— le parcours d'impression d'une couche ;
— la génération du G-code.

Toutes ces étapes ne sont pas exécutées strictement dans cet ordre et elles ne sont pas de fait une représentation fidèle de la structure du code de CuraEngine. Une description de la structure du code est détaillée dans [9]. De plus, afin d'améliorer les performances, CuraEngine exécute certaines de ces étapes selon un schéma producteur-consommateur (i.e. un thread produit des données qui sont consommées par un autre thread calculant une étape ultérieure en même temps).

Tranchage en couches (slicing) La première étape de l'algorithme est celle du découpage en tranches, permettant de convertir un maillage 3D en couches 2D. Pour générer chaque couche, cette étape détermine d'abord la hauteur à laquelle il faut produire la section transversale. L'intersection de cette section avec les triangles du maillage 3D forme des lignes qui sont ensuite assemblées pour former des polygones (chaque fois que cela est possible) dans le plan formant ainsi la couche.

Génération des zones par couche Cette étape divise chaque couche en zones. Une zone représente une partie à imprimer séparée des autres zones. Par exemple, si le modèle est une table, pour le plateau il n'y aura qu'une seule zone tandis qu'il y aura une zone différente pour chacun des pieds. Cette étape détermine également le types des parties à imprimer tel que les murs, la peau, le remplissage et les supports.

Remplissage des zones Pour chacune des zones d'une couche précédemment déterminée, cette étape traduit en lignes l'impression de chaque zone. Ces lignes sont déterminées en fonction du type de la partie à imprimer (mur, remplissage...) L'ordre d'impression de ces lignes (par exemple mur puis remplissage en fonction des paramètres utilisateurs) ainsi que le point de départ et d'arrivée des lignes sont également déterminés. La sortie de cette étape est un ensemble de couches ontenant les commandes de mouvement dans une représentation interne à CuraEngine.



```
info: using jerk mode
info: Built GCodeObject with 141085 instructions
Total displacement:            271911.487 mm
Total displacement extruding: 222437.360 mm  (81.8 %)
Average displacement:                1.954 mm     std deviation: 6.749 mm
Average displacement extruding:      1.600 mm     std deviation: 0.929 mm
Total extruded: 10773.098 mm
Total time:              194.49 min = 11669.3 s
Total time extruding:    177.18 min = 10630.74 s   (91.1 %)
Total time retracting:     0.38 min =   23.08 s   ( 0.2 %)
Number of retracts: 923
Extruding time per type:
                  WALL_INNER  81.06 min = 4863.7 s ( 45.8 % of extruding)
                  WALL_OUTER  81.09 min = 4865.6 s ( 45.8 % of extruding)
                        SKIN  15.02 min =  901.4 s (  8.5 % of extruding)
                        FILL   0.00 min =    0.0 s (  0.0 % of extruding)
                       SKIRT   0.00 min =    0.0 s (  0.0 % of extruding)
          SUPPORT_INTERFACE    0.00 min =    0.0 s (  0.0 % of extruding)
                     SUPPORT   0.00 min =    0.0 s (  0.0 % of extruding)
Volume: 25912.3 mm^3   Mass: 35.8 g
```

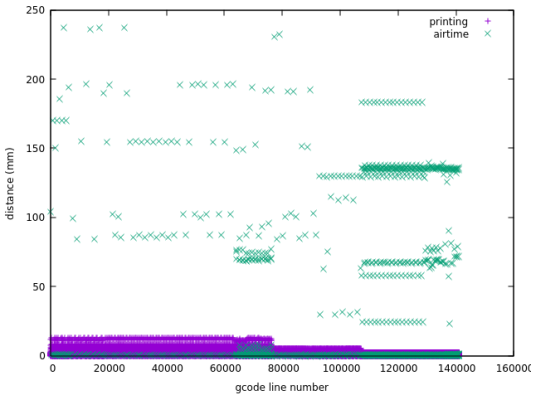


TABLE 3 – exemple modèle v30 avec 4 visières sur le plateau

Parcours d'impression d'une couche L'ordre de parcours pour l'impression d'une couche s'apparente à un problème du voyageur de commerce pour déterminer dans quel ordre imprimer les lignes. L'heuristique utilisée dans CuraEngine est celle du plus proche voisin. Elle est utilisée pour déterminer l'ordre de parcours des zones ainsi que l'ordre de parcours à l'intérieur de chacune des zones. Les stratégies d'évitement (les rétractions, l'évitement des collisions et le décalage en Z) sont également déterminées lors de cette étape.

Génération du G-code Dans cette étape finale, la représentation interne de CuraEngine modélisant l'ensemble des déplacements est traduite en commandes G-code pour l'ensemble des couches. Cette étape est également en charge de l'ensemble des calculs volumétriques déterminant la quantité de matière à déposer lors de chaque déplacement.

Automatisation et exploration des paramètres Afin d'explorer les variations sur le fichier G-code généré pour un modèle 3D par CuraEngine en fonction des paramètres de l'imprimante et de ceux sélectionné par l'utilisateur, nous avons développé un générateur d'appel de CuraEngine sans passer par le frontend. Ce générateur, outre l'appel à CuraEngine, permet d'extraire automatiquement la configuration de l'imprimante et le profil d'impression.

5.3 GATO3D

GATO3D est l'acronyme de « G-code Analysis Transformation and Optimization for 3D printing ». C'est une bibliothèque qui fournit une abstraction du G-code ainsi qu'une API pour le manipuler aisément. En premier lieu, GATO3D lit un fichier au format G-code et en construit la représentation en mémoire. Cette représentation peut être retranscrite dans un fichier G-code à la fin de la manipulation. Dans sa version actuelle, GATO3D analyse les fichiers G-code générés par le logiciel Ultimaker Cura 4.4.1.

GATO3D construit une liste d'instructions, propose des itérateurs pour la parcourir, et un mécanisme d'attributs pour attacher les résultats de diverses analyses aux instructions. Un certain nombre d'analyses et de transformations du G-code ont été réalisées pour les besoins de cette étude.

GATO3D nous permet grâce à ses analyses de caractériser les modèles à imprimer de façon textuelle et graphique afin de mieux identifier les opportunités d'optimisations. Dans l'exemple ci-dessous les croix vertes représentent la distance de chaque déplacement à vide.

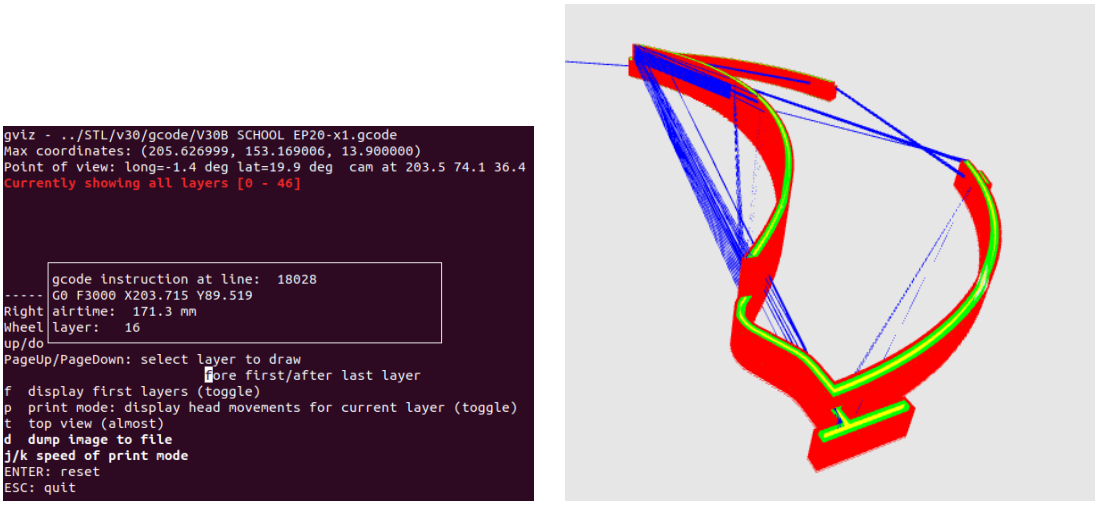


TABLE 4 – exemple modèle v30 avec 1 visière sur le plateau

GATO3D a permis de construire Gviz : une interface graphique qui permet de visualiser un modèle 3D au format G-code, de représenter graphiquement les propriétés des différents segments par couleur, ainsi qu'une représentation couche par couche et une approche interactive pour l'analyse fine des différents segments.

Finalement, GATO3D nous a permis de réaliser et tester la faisabilité d'une première optimisation consistant à réduire les déplacements à vide. La mise en œuvre de cette optimisation repose sur un certain nombre d'hypothèses et fait abstraction de certaines contraintes physiques pour permettre d'effectuer des premiers tests lors de cette étude. Notamment, elle est applicable sur un seul modèle 3D non-disjoint sur le plateau et ne nécessitant pas de support, elle suppose que le mode des déplacements à vide choisi par l'utilisateur est sans évitement des parties déjà imprimées (sauf rétraction), elle ne considère pas le temps de refroidissement du plastique entre les couches et ne réalise pas de traitement particulier pour les segments en surplomb. Ces restrictions restent néanmoins relativement raisonnables pour cette étude de faisabilité portant majoritairement sur les visières de protections.

Cette optimisation s'applique sur chaque couche indépendamment de la suivante à l'exception qu'elle essaie de minimiser le déplacement à vide entre deux couches successives en terminant le parcours d'une couche le plus proche possible du début de la couche suivante. Il est à noter que le point de départ d'une couche est déterminé par le trancheur en fonction des choix de l'utilisateur afin par exemple de masquer du mieux possible la jointure en plaçant celle-ci dans l'angle le plus aigu.

Pour chaque couche à imprimer un problème de voyageur de commerce est modélisé. Chaque ville représente dans le cas présent un segment à imprimer et la distance entre ces segments représentent les déplacements à vide. Ce problème est asymétrique dû à la notion de segment à la place de point. La notion de précédence est dans la version actuelle implicite avec la résolution du problème par couche à imprimer. La ville de départ est potentiellement différente de celle d'arrivée. Une fois le problème résolu, le G-code est généré afin de refléter le nouveau parcours déterminé et remplacer l'ensemble de la couche initialement déterminée par le trancheur.

La résolution est effectuée par un solveur que nous avons développé en utilisant la bibliothèque OR-Tools [14] de Google distribuée sous licence Apache License 2.0. Cette bibliothèque permet notamment de faire la résolution en utilisant différentes heuristiques de la littérature. Nous

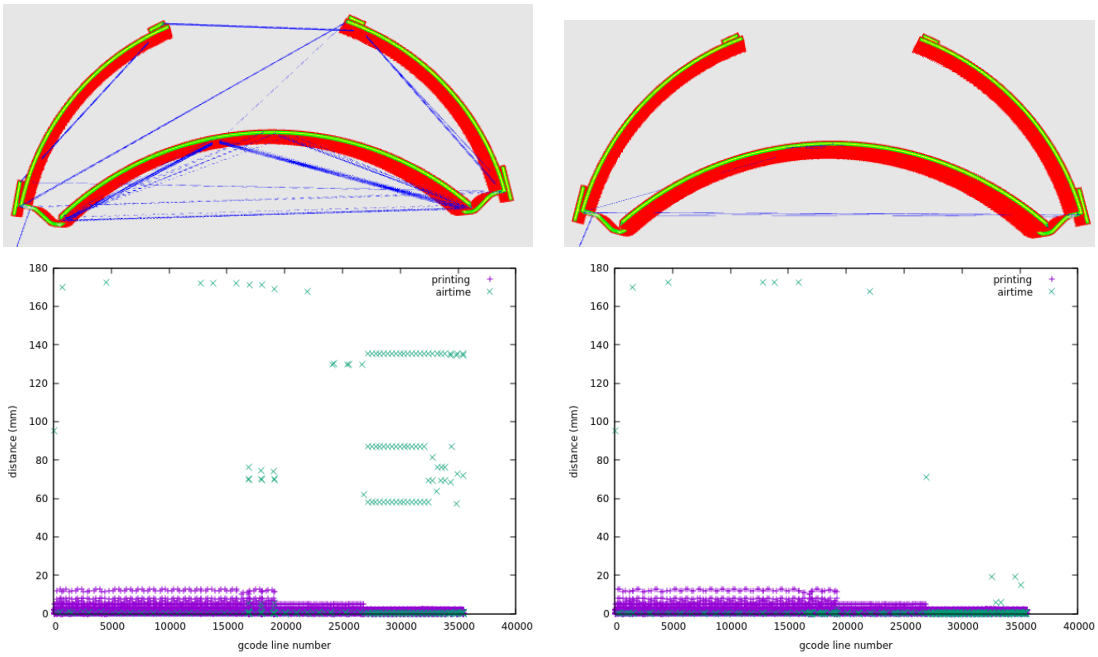


Table 5 – exemple modèle v30 avec 1 visière sur le plateau (version de référence à gauche et optimisée à droite)

avons pour cette étude retenu l'heuristique de Christofides [17] celui-ci ayant déjà montré des bons résultats pour ce type de problème [20]. L'exemple ci-dessous illustre le résultat après optimisation d'une visière v30 par rapport à la version générée avec Ultimaker Cura 4.4.1. En termes de temps d'impression, (vitesses toutes fixées à 50mm/s, soit toutes divisé par 2 par rapport au partenaire professionnel de cette étude et paramètres d'impression semblables) le modèle de référence prend 24min11s tandis que le modèle optimisé prend 21min17s soit un gain d'approximativement 12 %. La réduction en termes de déplacement à vide est également dans ce cas d'approximativement 12 % comme illustrée sur la Figure 1. Le gain en termes de temps d'impression n'est pas nécessairement équivalent au gain des déplacements à vide car d'autres facteurs entrent en ligne de compte (par exemple des vitesses de déplacement différentes selon la longueur des segments).

Pour les autres modèles 3D, nous observons également une réduction potentiellement significative sur certains modèles comme illustrée Figure 1. Ce gain est fortement dépendant de la forme du modèle 3D. Des déplacements à vide sont toujours présents sur la version optimisée. Parmi eux certains sont nécessaires tandis que d'autres sont optimisables en améliorant la modélisation et les contraintes sur le parcours. D'un autre côté, la prise en compte principalement des contraintes physiques mais aussi de certains paramètres utilisateurs pourrait avoir un impact négatif mais difficilement quantifiable à l'heure actuelle.

Comme indiqué précédent, le gain en termes de déplacement à vide ne se reflète pas dans les mêmes mesures sur le temps d'impression. Globalement le gain est tout de même significatif pour certains modèles comme indiqué dans le tableau 2. Dans la mise en œuvre actuelle de l'optimisation, définie pour des visières rapidement imprimables, certaines décisions de l'heuristique concernant le remplissage et la peau peuvent cependant avoir un effet négatif. C'est le cas pour le crochet et le support lunettes où le temps d'impression augmente. Nous comprenons les

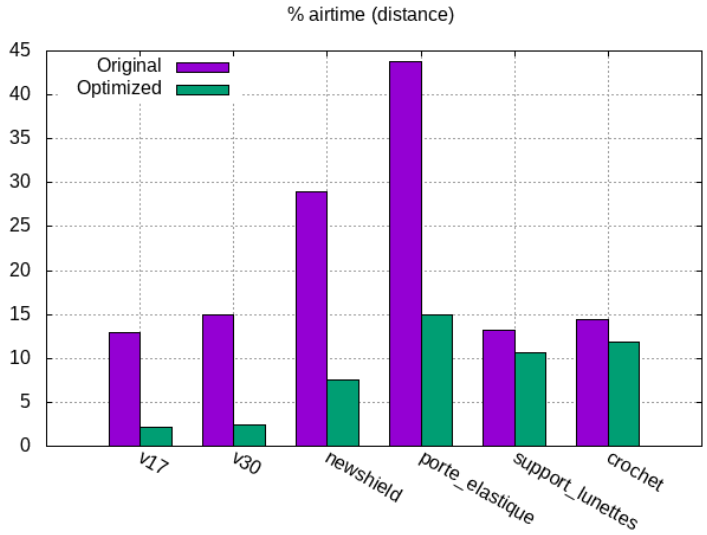


FIGURE 1 – Comparaison de la version non-optimisée et optimisée du pourcentage des déplacements à vide par rapport à l'ensemble des déplacements pour les différents modèles

Modèle 3D	Temps d'impression de référence	Temps d'impression après optimisation	Gain
v17	46min44s	42min44s	8 %
v30	24min11s	21min17s	12 %
newshield	1h7min46s	56min20s	16 %
porte élastique	19min43s	15min06s	23 %
support lunettes	21min52s	23min09s	-6 %
crochet	31min37s	32min14s	-2 %

TABLE 6 – Temps d'impression mesuré (Vitesse 50 mm/s ; buse 0.4 mm)

comportements amenant à cet effet. Ils sont liés à la façon dont l'heuristique ordonnance les petits déplacements ainsi qu'au point de départ d'une couche déterminé dans notre cas par le trancheur. Ces cas feront l'objet d'améliorations futures de notre mise en œuvre.

Pour ces deux modèles, nous avons également fait une estimation du temps d'impression avec le simulateur [11] utilisé dans [20] qui ne détecte pas ce comportement d'où la nécessité de faire des impressions réelles.

Nous nous sommes également intéressés à la problématique de passage à l'échelle de notre approche bien que la mise en œuvre actuelle n'ait pas été axée pour gérer ce problème. Il en résulte que l'heuristique est en mesure de traiter des problèmes avoisinant les 30K déplacements par couche. Cependant le temps de traitement par couche est dans ce cas important et nécessitera d'être réduit par la suite.

5.4 D'autres pistes d'optimisations

Déplacements en extrusion. Outre les déplacements à vide, les déplacements en extrusion pourraient également être minimisés. Ceci pourrait être réalisé en faisant un remplissage adaptatif

tant du point de vue de la densité que de la hauteur des couches pour le remplissage ainsi que les supports. Des tous premiers tests ont été réalisés dans ce sens afin de valider la possibilité de ce genre d'approche. Ces approches ont déjà été envisagées dans des articles [24, 23] il y a approximativement vingt ans. Elles ne sont néanmoins que peu exploitées à notre connaissance dans les trancheurs actuels. Ils nous apparaît intéressant de revisiter ces méthodes en prenant en compte les progrès tant au niveau des trancheurs que des imprimantes 3D. Une extension de ce type d'approche viserait à optimiser les impressions en vrai 3D [15, 19] mais cela nécessite d'étudier plus en détail leur mise en œuvre et l'impact sur les commandes G-code.

Prise en compte des contraintes utilisateur. Une meilleure prise en compte des contraintes spécifique à chaque modèle 3D ouvrirait des opportunités vis-à-vis des optimisations sur les déplacements (à vide et en extrusion). Ces informations peuvent être fournies par l'utilisateur quand celui-ci a une bonne connaissance du modèle 3D. Cette approche est évoquée comme challenge ouvert dans [22] et certains aspects sont disponible dans IceSL comme la possibilité de choisir le point de départ d'une couche. Ce type de contraintes pourrait être fourni par un visualiseur graphique comme Gviz. Dans le cas des visières, cela consisterait à indiquer la surface devant impérativement avoir une qualité élevée, en l'occurrence la partie en contact avec le front.

Extension et meilleure exploitation du firmware. Une autre piste d'optimisation réside dans l'utilisation et éventuellement la définition de nouvelles commandes G-code dans le firmware de l'imprimante. Cette direction est envisagée par les développeurs du firmware Marlin [12] avec des commandes permettant par exemple d'imprimer directement des arcs. Pour utiliser ces commandes, la problématique principale est d'être capable d'identifier de tels mouvements à partir du G-code constitué exclusivement de segments. La définition de nouvelles commandes permettant entre autres de grouper des commandes ou d'avoir une gestion plus fine de la vitesse de chacun des moteurs permettrait potentiellement d'accélérer les déplacements en fonction des directions. Une partie de cette approche peut également se faire sur le G-code directement ce qui nous a permis de réaliser des tous premiers tests dans ce sens. Ce genre d'approche pourrait être combiné avec les méthodes visant à réduire les secousses [18, 13] lors de l'impression.

Optimisation du firmware. Concernant le firmware, il semblerait que pour certains types de modèles à imprimer, le temps de traitement du firmware soit sur le chemin critique. Analyser plus en détail ces comportements et éventuellement optimiser le firmware est envisageable et pleinement dans nos thématiques de recherche.

Il est à noter que différents modèles d'imprimantes par dépôt de filament, voire également leurs configurations, pourraient révéler différents comportements vis-à-vis de ces optimisations.

6 Bilan

Par rapport aux objectifs initiaux, certaines adaptations ont été faites dû à plusieurs facteurs : la situation particulière, le matériel à disposition ainsi que les logiciels associés. Il en résulte néanmoins que cette étude montre un bon potentiel concernant les possibilités d'optimisations pour réduire le temps d'impression.

Concernant les impressions réalisées lors de cette étude, elles ont été grandement minimisées par rapport à ce qui était initialement prévu. Au début de l'étude, les tests d'observations des comportements d'impressions afin d'évaluer et de comprendre les opportunités d'optimisations ont été drastiquement réduits grâce au développement de GATO3D. GATO3D nous a permis

d'avoir rapidement des statistiques sur les déplacements à vide comme leur distance et leur localisation dans les fichiers G-code.

Une analyse du trancheur Ultimaker Cura 4.4.1 a été menée et présentée en section 5. Une brève analyse des comportements des principaux trancheurs existants a également été effectuée pour les déplacements à vide et il en résulte que les comportements observés sont présents dans l'ensemble de ces trancheurs. Vers la fin de l'étude, il était envisagé la mise en œuvre d'optimisations à l'intérieur du code de Ultimaker cura. Cette piste n'a pas été poursuivie à la vue des travaux existants [21, 20]. Appliquer les optimisations sur le G-code directement a certains avantages : la portabilité pour d'autres trancheurs, un plus grand degré de liberté pour les optimisations n'étant pas limité par la structure et l'ordre imposé pour l'impression des différentes parties (mur et remplissage) par un logiciel lors de la génération du G-code. La contrepartie étant d'avoir à reconstruire et retrouver les informations non transmises dans le fichier G-code.

La mise en œuvre de l'optimisation portant sur les déplacements à vide est basée sur un certain nombre d'hypothèses simplificatrices pour réaliser les tests de cette étude portant principalement sur les visières de protection. Pour un usage plus généraliste, ces limitations devront être levées ainsi que la problématique classique du passage à l'échelle. Outre montrer la faisabilité et obtenir une estimation du gain, ces tests ont permis d'observer différents comportements en fonction de l'heuristique choisie et des opportunités d'améliorations de cette optimisation. Concernant les limitations de cette optimisation, il existe des modèles imprimables avec très peu voire sans déplacements à vide comme les modèles en spirale. Également, le ratio entre la vitesse de déplacement possible en extrusion et à vide des moteurs impacte fortement le gain. Dans cette étude nous nous sommes placés dans le cas où cette vitesse est identique ce qui correspond aux paramètres utilisés par le partenaire professionnel pour l'impression de visières sur leur imprimante. Néanmoins ceci a pour effet de maximiser le gain. Ce type de paramètres n'est pas nécessairement réaliste pour des imprimantes grand public. Par rapport aux travaux existants [21, 20], nous nous différencions en optimisant sur le G-code directement ce qui permet de ne pas être contraints par l'ordre d'impression imposé par Ultimaker Cura. Il faudrait mener une étude comparative précise pour estimer et quantifier le gain atteignable avec ce degré de liberté supplémentaire. Dans tous les cas, notre approche permet d'être applicable à d'autres trancheurs. Nous avons également montré, par un contre-exemple, que leur méthode d'évaluation basée sur un simulateur d'impression n'est pas fiable.

7 Perspectives

Outre les autres pistes d'optimisations évoquées précédemment, d'autres problèmes sous-jacents en lien avec les recherches menées en compilation par l'équipe PACAP apparaissent intéressants. Le fichier G-code a de bonnes propriétés en termes de techniques usuelles de compilation : la suite d'instructions peut être considérée comme un seul bloc de base et de fait le théorème de Rice ne s'applique pas dans ce cas. Cependant, la problématique d'une représentation intermédiaire efficace pour manipuler les instructions en fonctions des contraintes de précédences, limites physiques, paramètres du trancheur et les évolutions des techniques d'impressions 3D ne nous semble pas trivial. Une bonne représentation permettrait de manipuler et d'optimiser probablement plus rapidement le G-code.

Il y a également la perte d'information au niveau du G-code par rapport au format STL et de façon plus générale également entre le design initial et le format STL. D'autres formats que le STL existent, cependant ils ne semblent pas le remplacer à l'heure actuelle. Cette problématique de perte d'informations est mentionnée comme un challenge ouvert dans [22]. L'applicabilité

des méthodes utilisées traditionnellement en compilation pour traiter cette problématique nous semble intéressante à étudier.

L'estimation du temps d'impression s'est avéré être une problématique complexe. Les différents trancheurs et simulateurs que nous avons essayés ne permettent pas d'obtenir une estimation précise voire dans certains cas ils peuvent amener à des conclusions erronées. Un modèle précis permettrait de réduire considérablement le temps passé à faire des impressions réelles. Définir un modèle précis pourrait se faire en analysant (éventuellement en ligne) voire en simulant le firmware de l'imprimante. Une approche complémentaire à celle-ci serait d'utiliser des capteurs pour comprendre et déduire les phénomènes du déplacement de la tête d'impression. Ce genre de technique est employée dans d'autres domaines de recherche du laboratoire comme la capture de mouvement humain.

Références

[1] `https://3doptimizer.com/`.

[2] `https://cults3d.com/fr/mod%C3%A8le-3d/outil/shield-covid-v30-yann-vodable-feuille-perforee-standard`.

[3] `https://covid3d.org/projects/porte-elastique-pour-masque-petit-modele`.

[4] `http://rdm.cnrs.fr/spip.php?article560`.

[5] `https://www.myminifactory.com/fr/object/3d-print-115247`.

[6] `https://www.thingiverse.com/thing:4238879`.

[7] `https://covid3d.org/projects/lunettes-de-protection`.

[8] `https://ultimaker.com/fr/software/ultimaker-cura`.

[9] `https://github.com/Ultimaker/CuraEngine`.

[10] `https://support.ultimaker.com/hc/en-us/sections/360003548619-Print-settings`.

[11] `https://www.thingiverse.com/thing:44286`.

[12] `https://marlinfw.org`.

[13] `https://hackaday.io/project/7045-splinetravel`.

[14] OR-Tools. `https://developers.google.com/optimization/`.

[15] Daniel Ahlers. 3D printing of nonplanar layers for smooth surface generation. Master's thesis, Universität Hamburg, 2018.

[16] Kenneth Castelino, Roshan D'Souza, and Paul K Wright. Toolpath optimization for minimizing airtime during machining. *Journal of manufacturing systems*, 22(3) :173–180, 2003.

[17] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, 1976.

[18] Molong Duan, Deokkyun Yoon, and Chinedum E Okwudire. A limited-preview filtered b-spline approach to tracking control–with application to vibration-induced error compensation of a 3D printer. *Mechatronics*, 56 :287–296, 2018.

[19] Jimmy Etienne, Nicolas Ray, Daniele Panozzo, Samuel Hornus, Charlie CL Wang, Jonàs Martínez, Sara McMains, Marc Alexa, Brian Wyvill, and Sylvain Lefebvre. Curvislicer : slightly curved slicing for 3-axis printers. *ACM Transactions on Graphics (TOG)*, 38(4) :1–11, 2019.

[20] Kai-Yin Fok, Nuwan Ganganath, Chi-Tsun Cheng, and K Tse Chi. A 3D printing path optimizer based on Christofides algorithm. In *2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, pages 1–2. IEEE, 2016.

[21] Nuwan Ganganath, Chi-Tsun Cheng, Kai-Yin Fok, and K Tse Chi. Trajectory planning for 3D printing : A revisit to traveling salesman problem. In *2016 2nd International Conference on Control, Automation and Robotics (ICCAR)*, pages 287–290. IEEE, 2016.

[22] Marco Livesu, Stefano Ellero, Jonàs Martínez, Sylvain Lefebvre, and Marco Attene. From 3D models to 3D prints : an overview of the processing pipeline. In *Computer Graphics Forum*, volume 36, pages 537–564. Wiley Online Library, 2017.

[23] Ka Mani, Prashant Kulkarni, and Debasish Dutta. Region-based adaptive slicing. *Computer-Aided Design*, 31(5) :317–333, 1999.

[24] Emmanuel Sabourin, Scott A Houser, and Jan Helge Bøhn. Accurate exterior, fast interior layered manufacturing. *Rapid Prototyping Journal*, 1997.

[25] Pang King Wah, Katta G Murty, Ajay Joneja, and Leung Chi Chiu. Tool path optimization in layered manufacturing. *IIE Transactions*, 34(4) :335–347, 2002.

[26] Yang Weidong. Optimal path planning in rapid prototyping based on genetic algorithm. In *2009 Chinese Control and Decision Conference*, pages 5068–5072. IEEE, 2009.

A Illustration des déplacements à vide présents dans les trancheurs

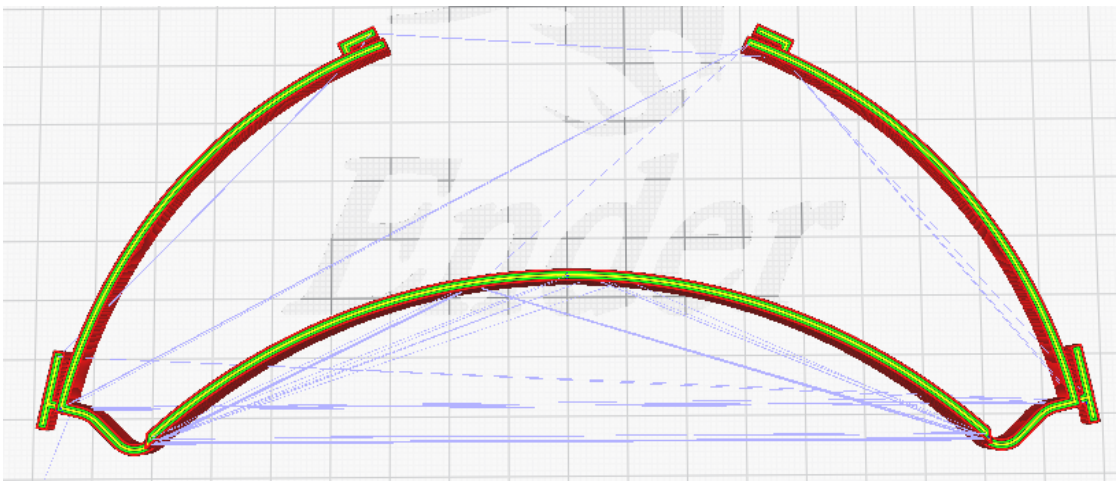


FIGURE 2 – Visière V30 - Ultimaker Cura 4.4.1 - Déplacements à vide en bleu

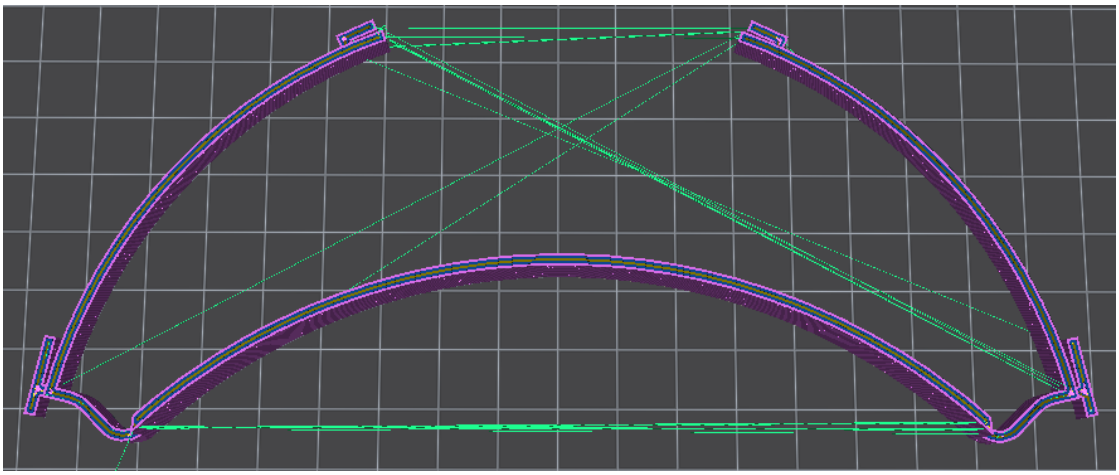


FIGURE 3 – Visière V30 - IceSL 2.3.4 - Déplacements à vide en vert

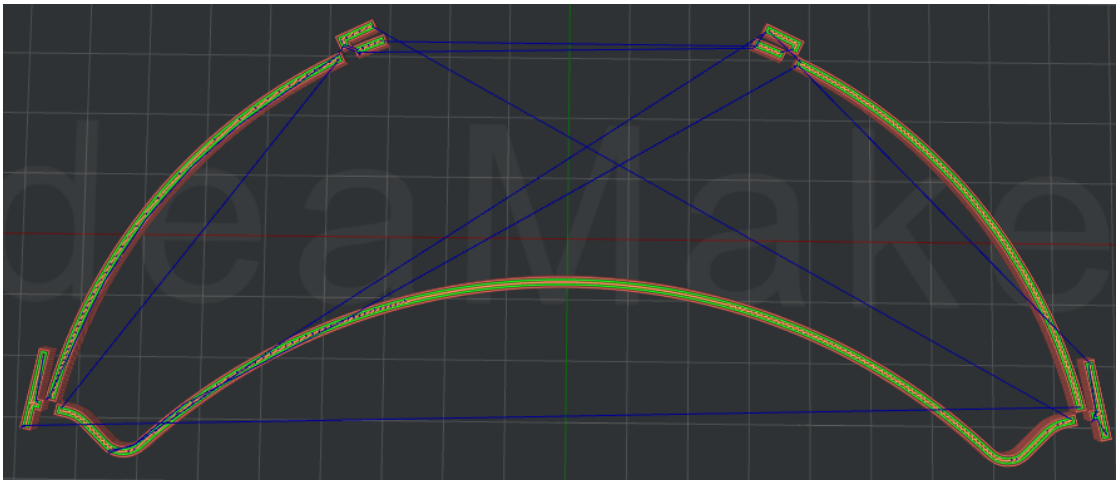


FIGURE 4 – Visière V30 - IdeaMaker 3.5.3.4250 - Déplacements à vide en bleu

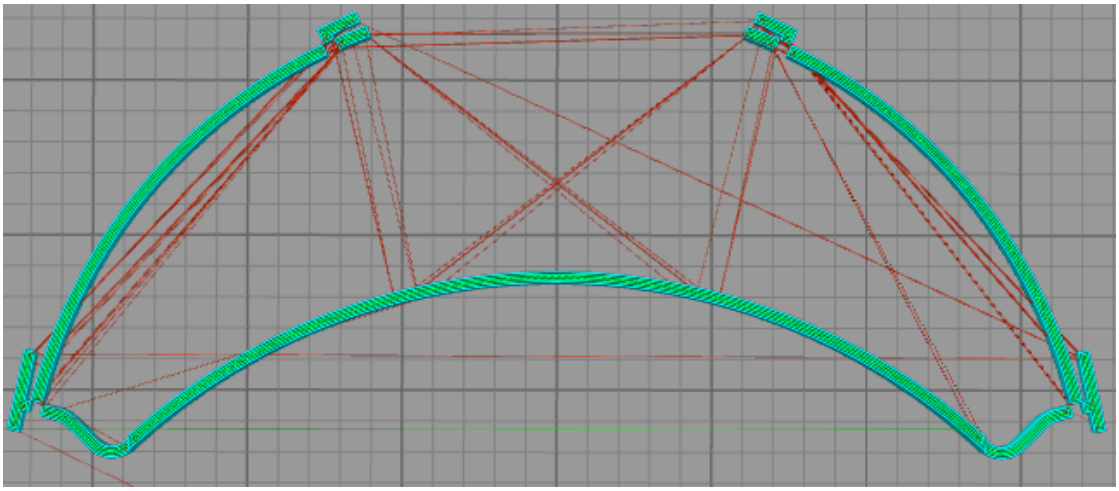


FIGURE 5 – Visière V30 - Simplify3D 4.1.2 - Déplacements à vide en rouge

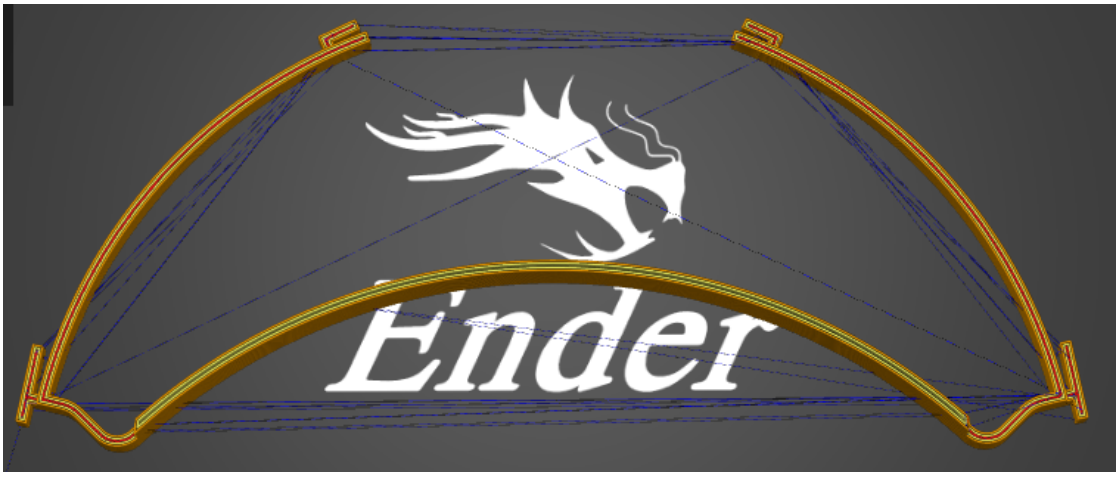


FIGURE 6 – Visière V30 - PrusaSlicer 2.2.0 - Déplacements à vide en bleu

RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-0803