Petit voyage initiatique dans le monde de l'impression 3D

INF574 – Rapport sur le projet de fin de trimestre

Cédric JAVAULT – Master AI Year 1

Ma première découverte de l'informatique, dans les années 80, fut presque totalement autodidacte ; elle m'a amené à de nombreuses expériences, depuis le fer à souder pour assembler les composants jusqu'aux détails du langage machine (j'ai même écrit et commercialisé une « méthode » pour apprendre ce langage sur Atari ST), en passant par la signature d'un contrat avec Ubi Soft pour un jeu… qui ne sera jamais finalisé.

C'est à l'X où je suis admis en 1992, que j'ai fait ma seconde découverte de l'informatique : avec le langage C, les algorithmes, les structures de données, je me plongeais dans mon projet informatique (faire un jeu d'Othello qui joue raisonnablement bien), décortiquait la théorie du codage RSA et les conséquences vertigineuses si P=NP, programmais en majeure de seconde année un Tetris en paramétrant directement les transistors sur Xilinx et en créant le jeu d'instruction et le programme adéquat.

Après un début de carrière raisonnablement scientifique (contrôle de la sûreté nucléaire), j'ai quitté la fonction publique pour créer ma petite entreprise : microscopique organisateur de séjours de vacances scientifiques pour enfants au tournant des années 2000, nous étions devenus le numéro 2 français des colonies de vacances lors du rachat par l'UCPA en 2017… Mon principal contact avec l'informatique depuis plus de 20 ans s'est résumé à la création de toute pièce un gros fichier ACCESS pour gérer toute l'activité !

Après avoir accompagné un temps mon acheteur, j'ai décidé de retourner en formation pour préparer la suite de ma carrière professionnelle : début septembre 2019, de retour à l'X dans le cadre d'un master Artificial Intelligence & Advanced Visual Computing pour une grosse remise à niveau, je fais donc ma troisième découverte de l'informatique : il y a 3 mois, je ne connaissais rien à la programmation orientée objet, n'avais jamais écrit une ligne en Python, et ignorait jusqu'à l'existence des triangles meshes !

Cette longue introduction me permet d'en venir au fait : j'ai choisi un projet autour de la 3D… parce que je n'y connaissais rien et que cela m'a toujours fasciné. J'ai plus précisément choisi l'article "Spin It"… just because it looked so funny! Je n'ai pas été déçu, j'ai appris énormément de choses, et même si je n'ai pas pu – pour l'instant ! – mener ce projet à son terme, je pense avoir fait un vrai petit voyage de découverte dans le monde de l'impression 3D !

1. Roadmap de mon projet

Je me suis basé sur l'article « Spin It » de 2014 de Moritz Bächer & all (Disney Research Zurich). L'idée principale du papier est de partir d'un objet (plein) qui n'a pas les qualités intrinsèques pour tourner comme une toupie, de modifier sa répartition de masse interne en créant du vide… pour fabriquer un objet qui tourne très bien.

La vidéo présentée par les auteurs est assez spectaculaire ; elle montre également les idées connexes développées par les auteurs, idées auxquelles je ne me suis pas particulièrement intéressé par manque de temps : possibilité d'optimiser également un yo-yo, d'aller plus loin que les simples creusements en déformant la pièce, de mixer plusieurs matériaux pour bénéficier de densité et donc d'effets plus forts…

Pour ce projet, je me suis défini dès le départ une 'roadmap' des différentes étapes pour réussir le projet… ou au moins avoir des choses à présenter si je n'arrivais pas à tout implémenter. La voici :

1. *(Faire la roadmap)*
2. Comprendre le papier en détail
3. Trouver une toupie 3D et d'autres objets qui, eux, ne tournent pas
4. Imprimer en 3D ces objets pour évaluer dès ce stade d'éventuelles difficultés pratiques

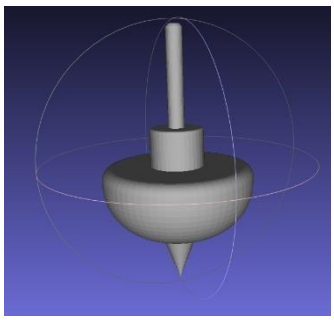
5. Implémenter les calculs du tenseur d'inertie et les vérifier sur un modèle simplifié
6. Faire une optimisation « à la main » pour calculer une forme « creusée » qui devrait tourner
7. Imprimer en 3D cette forme « optimisée à la main » / vérifier la pertinence de mon travail
8. Implémenter l'étape d'optimisation sur le modèle simplifié
9. Implémenter l'algorithme complet (avec split & merge) et optimiser un objet 3D
10. Imprimer en 3D cette forme « optimisée par l'algorithme »
11. *(Rédiger le rapport !)*

2. Les idées clés de l'article 'Spin It'

Voici ce qui me semble être un résumé décent des points clé de l'article :

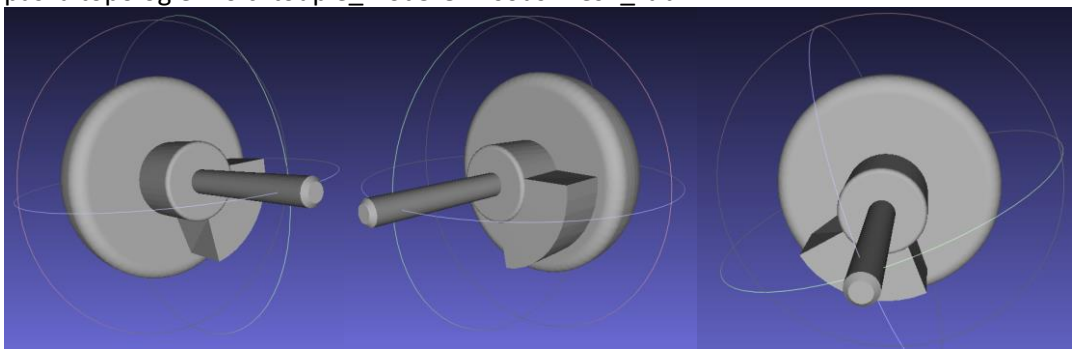
- L'utilisateur doit donner : une forme pleine sans frontière (donc concrètement un ensemble de points V et de triangles orientés N -> pour moi, ce sera un fichier OFF). Il doit aussi indiquer un axe de rotation et le point sur lequel il souhaite qu'il y ait rotation. Concrètement, cela veut dire qu'après une translation et une rotation, on se ramène à : Point de contact = O (0,0,0) et Axe de rotation=Oz (donc z>=0 pour tous les points V).
- Petit (gros pour moi !) rappel de mécanique du solide :
    - Le moment d'inertie : en translation, la masse mesure la résistance au changement de vitesse (d'où $E_{cinétique\ translation} = ½\ m\ V^2$). En rotation, c'est le moment d'inertie qui exprime la résistance au changement de vitesse angulaire ($E_{cinétique\ de\ rotation} = ½\ J\ \omega^2$).
    - Il y a un moment d'inertie par axe de rotation… et tout objet solide rigide se comporte en rotation comme un ellipsoid : son moment d'inertie peut s'exprimer par une matrice 3*3 qui est diagonalisable en base orthonormée et dont les valeurs propres $I_c >= I_b >= I_a$ sont positives. Dit autrement, tout objet a 3 axes orthogonaux de rotation privilégiée.
    - Pour tourner sur elle-même, une « pseudo toupie » doit respecter quelques conditions bien détaillées dans l'article. Par exemple : centre de masse sur Oz (!), $I_c >> I_b$ et I_a, certains termes de la matrice d'inertie doivent être nuls…
    - Et on aboutit à une « énergie » à minimiser sous différentes contraintes !
- L'algorithme propose de discrétiser la forme 3D sous la forme d'un octree :
    - Dans un octree, toutes les nœuds représentent des cubes (H=L=l) autour d'un point ; la racine est centrée au milieu de l'objet et le comprend entièrement (taille = Max(dimX, dimY, dimZ)) ; chaque nœud peut être une feuille ou avoir 8 fils qui sont alors les 2 x 2 x 2 sous-cubes de taille la moitié de la taille du père.
    - Pour notre problème d'optimisation du remplissage, l'objet est approximé par un octree dense au niveau des frontières, moins denses ailleurs ; dans chaque feuille de l'octree, la densité est constante (comprise entre 0 et 1)
    - Cette structure de données est adaptée au problème car en pratique, l'algorithme d'optimisation concentre la matière dans certaines parties de l'objet et vide les autres : cela permet de procéder par itération : à la fin de chaque cycle d'optimisation continue (densité = valeur réelle entre 0 et 1), les cellules de densité 0 ou 1 (à un seuil près) sont figées et rassemblées ; les autres sont subdivisées en 8 cellules pour continuer et aboutir à la fin (à quelques cellules frontière prêt) à une densité discrète 0 et 1 partout !
    - NB1 : il y a des contraintes de fabrication sur les objets réels (ils ont une épaisseur extérieure : ne pas laisser l'algorithme vider les bords !)
    - NB2 : Il est possible, grâce au théorème de la divergence, de faire les calculs sur les surfaces et pas sur les volumes (calculs exacts et plus rapides)
- L'optimisation consiste à minimiser une énergie fonction des densités des cellules actives sous contrainte ; différents algorithmes d'optimisation sont évoqués dans l'article (en pratique, l'optimisation m'a posé un tel problème que je n'ai pas réussi à le surmonter faute d'algorithme coopératif : je vais y revenir en détail).

3. Trouver une toupie 3D et d'autres objets qui, eux, ne tournent pas



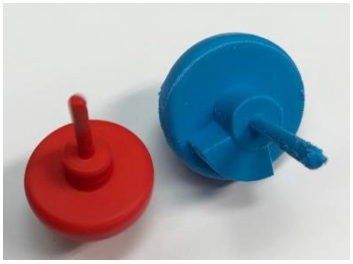
Mon voyage au pays de l'impression 3D a en pratique commencé par une exploration sur Internet. En fait, trouver l'objet 3D qu'on cherche est compliqué : les sites « gratuits » sont en réalité payant à 99%, leurs moteurs de recherches ne sont pas très performants. J'ai tout de même réussi à trouver une belle toupie 3D (2787 points et 5570 faces – voir ci-contre). En revanche, évidemment, la requête « toupie 3D qui ne toupite pas » ne donne pas de résultat… Bizarre, non ?

Il m'a donc fallu fabriquer mes « peudos toupies cassées » moi-même. Ce fut une excellente occasion de vérifier que j'avais compris le format des fichiers OFF puisque j'ai dû « bricoler » à la main dedans. Concrètement, j'ai « opéré » la vraie toupie 3D sous Excel pour la déformer : deux rangées de points (30 points en tout) ont été écartées de l'axe. Je n'ai modifié que la géométrie (les coordonnées des points) et pas la topologie. Voici toupie_mod5.OFF sous Mesh_Lab:



4. Imprimer en 3D ces objets pour évaluer d'éventuelles difficultés concrètes dès ce stade

La partie amusante du projet, c'était de jouer avec une imprimante 3D « à moi ». Mon choix s'est porté sur un modèle d'entrée_de_gamme_mais_pas_complètement_le_moins_cher : 499 € pour une imprimante 3D française Dagoma. Bon, ce n'est pas le sujet, mais vu les difficultés depuis une semaine, je ne suis pas parti pour la recommander… mais attendons quelques pages !



Pour l'instant, en ce début de projet, tout va bien : l'imprimante s'installe facilement, et m'imprime la toupie et la toupie modifiée gentiment. On peut modifier la taille et bien sûr la couleur en changeant de filament. Et comme prévu, la toupie 3D spin admirablement bien, toupie_mod5 pas du tout (en tous cas, pas sur l'axe Oz) !

En revanche, il y a bien une petite surprise… qui justifie sans doute a posteriori d'avoir imprimé en 3D tôt dans le projet : la machine que j'ai achetée ne fabrique pas des objets pleins ! Pour l'intérieur, elle propose seulement trois modes : vide (0%), rempli (17% de matière) et renforcé (33% de matière). En théorie, cela n'est qu'un paramètre mineur mais …

- Est-ce vraiment 33%, un peu plus, un peu moins ?
- Les objets sont fabriqués avec une épaisseur manifestement pleine (100% de matière) d'environ 1,5mm ; par construction ce 100% de matière est sur les bords de l'objet, avec un poids maximal dans le tenseur d'inertie : mais est-ce 1,2mm 1,5mm 2mm ? Rien dans la doc…
- Ces deux inconnues ne vont-elles pas ruiner mon optimisation si je n'ai pas les valeurs exactes ?

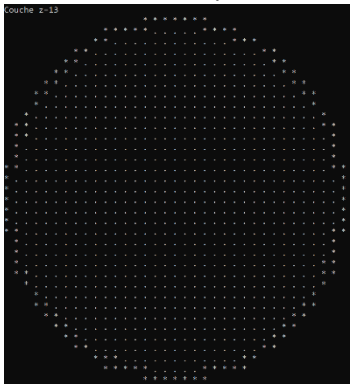
En résumé, petite découverte du jour : il y a les objets théoriques sur lesquels on va faire de beaux calculs… et les objets réels 3D avec les contraintes réelles de l'impression 3D : il va falloir un peu d'imagination et de persévérance pour combler le gap entre les deux mondes !

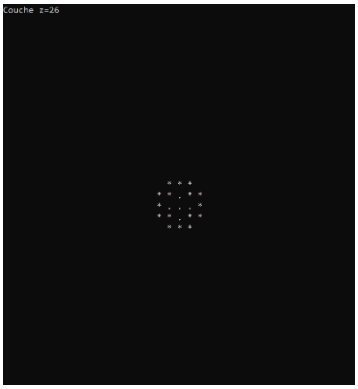
5. Implémenter le calcul du tenseur d'inertie et les vérifier sur un modèle simplifié

L'article « Spin It » a une annexe qui détaille l'utilisation du théorème de la divergence et propose un algorithme clé en main pour calculer la masse, le centre de gravité et le tenseur d'inertie à partir de V et N. Implémenter cet « algorithme 1 » était donc simple ; je l'ai écrit en C++ (puis en Python – voir plus loin) ; mais il m'était impossible d'évaluer la pertinence des résultats pour le tenseur d'inertie sans un autre outil (la masse et le centre de gravité, passe encore, mais le tenseur d'inertie…).

J'ai donc programmé en C++ un petit modèle sans ambition autre que de vérifier mes calculs ; il me servira plus tard à tenter d'implémenter l'optimisation. Ce modèle consiste à diviser l'espace en MAX^3 cellules (MAX paramétrable), et de les mettre à 0 (extérieur), 1 (intérieur) ou 2 (frontière). Pour calculer les cellules de la frontière, je parcours toutes les faces et met à 2 les voxels qui intersectent la face. Puis, je mets une « seed » à l'intérieur de la forme à 1 et la fait croitre jusqu'à ce que la frontière soit rencontrée.

Une petite visualisation par coupe (Z=cte) m'a permis de vérifier que l'implémentation était correcte avec '*' pour la frontière et '.' pour l'intérieur :

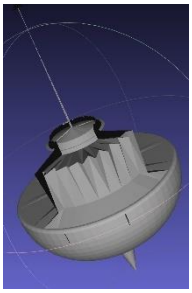




Le calcul de la masse, du centre de gravité, et du tenseur d'inertie étant très simples sur mon modèle simplifié, la probabilité d'un bug était beaucoup plus faible. En tout état de cause, j'ai eu des résultats cohérents entre « l'algorithme 1 » de l'article (résultat exact) et mon modèle simplifié : go on !

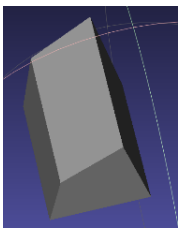
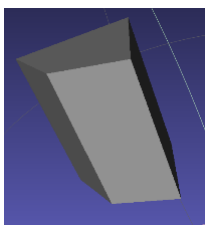
6. Faire une optimisation « à la main » pour calculer une forme « creusée » qui devrait tourner

Pour l'étape suivante, je voulais calculer un objet théorique qui respectait à peu près les contraintes de l'article ; l'idée était ensuite de le fabriquer pour de vrai et de vérifier que ma toupie_mod5 creusée spinait.



- J'ai tout d'abord calculé l'intérieur de la toupie modifiée en supposant que l'épaisseur était de 1.5mm : j'ai fait un petit programme qui prend un objet 3D, et bouge chaque point de 1.5mm vers l'intérieur le long de la normale. Cela donne un résultat un peu baroque sur certains points mais j'ai estimé que l'approximation était raisonnable. Mon « vrai » objet de départ est (Toupie_mod5) – 67% x (Intérieur de Toupie mod5) si le remplissage est bien de 33%.

- J'ai ensuite recodé l'algorithme 1 de l'article en Python (ce langage est plus facile à manipuler) et exprimé l'« erreur » que je voulais minimiser (erreur = somme normalisée des 5 quantités qui doivent être nulles dans les contraintes d'optimisation). Puis, après avoir un peu tâtonné (mais merci Blender) pour trouver une forme qui est entièrement contenue dans l'intérieur de Toupie Mod 5 et qui n'a que quelques degrés de liberté. J'ai abouti à Cube_tri_modifi3 qui a 6 degrés de liberté : Zmin et Zmax, distance à l'axe en haut et en bas, angle min et max…

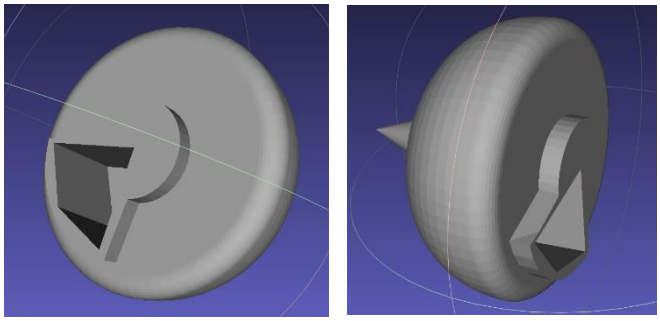
- … Et fait tourner mon algorithme en Python pour trouver la variante de Cube_tri_modifi3 parmi 10^6 = 1 million de possibilités, variante qui minimise le mieux mon erreur en faisant varier les 6 paramètres mais en restant dans l'enveloppe de la toupie modifiée. Au final, j'ai ramené l'erreur de 4.4 millions à 16 644.

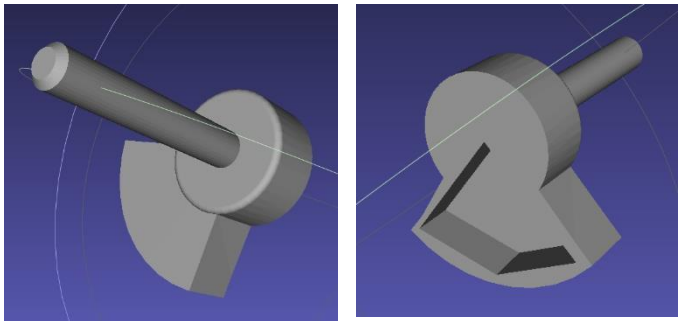
```
**********TOUPIE 5 PLEINE**********
La masse est de :  7635.715540686925
Position du centre de masse : [-0.39439748 -0.28486677 18.84894161]
Condition n°1 Sx=0 et là on a :  -3011.506958700893
Condition n°2 Sy=0 et là on a :  -2175.161620371006
Condition n°3 Sxz=0 et là on a :  -79082.75799457214
Condition n°4 Syz=0 et là on a :  -57079.34522320609
Condition n°5 : la valeur suivante doit être nulle :  -14383.14698574219
Note globale (0=parfait) 13951047.48724244
**********TOUPIE 5 MODIFIEE ET REMPLIE SEULEMENT A 33%**********
La masse est de :  4501.280234129203
Position du centre de masse : [-0.37667142 -0.26970472 19.1972059 ]
Condition n°1 Sx=0 et là on a :  -1695.5036344021084
Condition n°2 Sy=0 et là on a :  -1214.016543713641
Condition n°3 Sxz=0 et là on a :  -46656.86103105845
Condition n°4 Syz=0 et là on a :  -33389.868669380114
Condition n°5 : la valeur suivante doit être nulle :  -8738.85027693727
Note globale (0=parfait) 4437354.32265855
```

```
On a mieux avec 7 3 0 0 2 0    - nouvelle meilleure note :  17840.760793115332
On a mieux avec 8 0 1 1 2 0    - nouvelle meilleure note :  17458.115468959524
On a mieux avec 8 2 0 0 2 0    - nouvelle meilleure note :  17242.68789675037
On a mieux avec 9 1 0 0 2 0    - nouvelle meilleure note :  16644.6150003854
Meilleur résultat avec :
haut= 33.2
bas= 19.2
teta1= 3.0
teta2= 4.5
l1= 10.0
l2= 11.0
Note :  16644.6150003854
**********'CUBE' FINAL à 33% **********
La masse est de :  347.4585415342944
Position du centre de masse : [-4.98084986 -3.46941521 22.02205438]
Condition n°1 Sx=0 et là on a :  -1730.6388289313943
Condition n°2 Sy=0 et là on a :  -1205.4779492587036
Condition n°3 Sxz=0 et là on a :  -38303.761954357906
Condition n°4 Syz=0 et là on a :  -26680.51799007876
Condition n°5 : la valeur suivante doit être nulle :  -7370.5166616320885
Note globale (0=parfait) 4520642.638960066
**********TOUPIE 5 MODIFIEE ET REMPLIE SEULEMENT A 33% - 'CUBE' FINAL à 33%**********
La masse est de :  4153.821692594908
Position du centre de masse : [ 8.45852257e-03 -2.05559966e-03  1.89609132e+01]
Condition n°1 Sx=0 et là on a :  35.135194529285855
Condition n°2 Sy=0 et là on a :  -8.538594454937538
Condition n°3 Sxz=0 et là on a :  -8353.099076700542
Condition n°4 Syz=0 et là on a :  -6709.350679301355
Condition n°5 : la valeur suivante doit être nulle :  -274.77575450685606
Note globale (0=parfait) 16644.6150003854
```

7. Imprimer en 3D cette forme « optimisée à la main » pour voir d'éventuelles contraintes de fabrication sur le modèle simple

Pour vérifier la pertinence de mon « optimisation à la main », il fallait bien sûr imprimer l'objet théorique construit plus haut. Blender m'aidait à régler la première difficulté technique : calculer la toupie modifié 3D « moins » le cube modifié puis couper l'objet 3D en deux morceaux (indispensable pour l'impression). Je découvrais lors de l'impression une épaisseur de 2 x 1.5mm environ au niveau de la coupure… que j'avais oublié en prendre en compte dans les calculs (mais heureusement, c'est presque symétrique et proche de l'axe -> erreur faible).





La seconde difficulté provenait des inconnues évoquées plus haut sur l'épaisseur et sur le pourcentage de remplissage. Avec une balance précise (4.5g pour la toupie 3d de base), la densité du filament utilisé (1.24g/cm^3), la surface (calculée en Python avec partir de V et F), j'aurai dû pouvoir valider l'épaisseur et le taux de remplissage… mais cela ne collait pas bien. Hum…

J'ai donc décidé de contourner le problème en imprimant en 5 tailles différentes : taille 1 mais aussi 1.3 et 1.69 d'une part, 0.77 et 0.59 d'autre part. L'idée était que l'épaisseur serait toujours la même et que je pouvais ainsi ajuster le rapport épaisseur/taille de la pièce sans recommencer à zéro mais juste en imprimant des homothéties de ma toupie modifiée, creusée et coupée en deux. Et cela a marché :

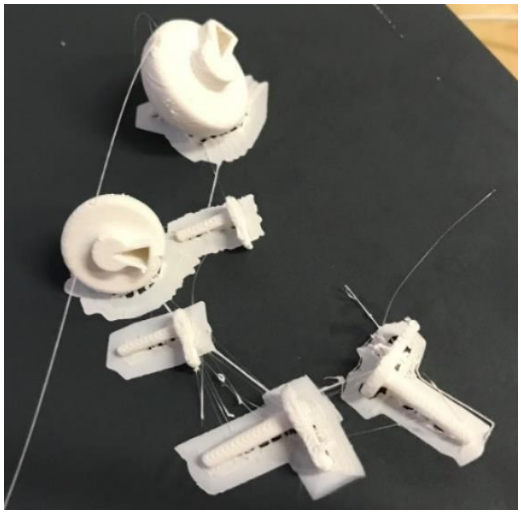
- D'une part,les toupies creusées tournent « moins mal » que les toupies non creusées
- Mieux : la taille 1.30 est suffisamment proche de l'optimum pour faire très correctement illusion (démonstration mercredi !).

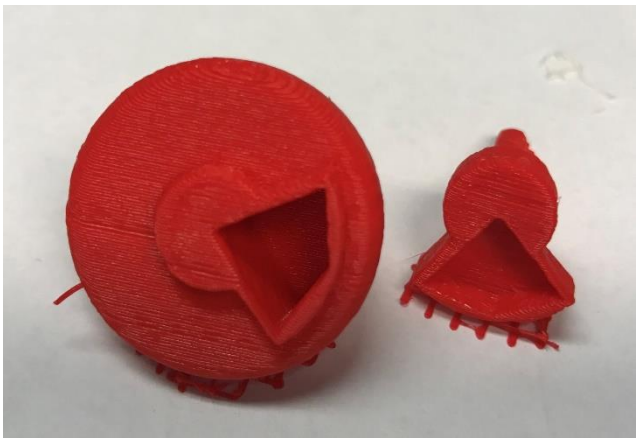
Pour mémoire, j'ai rencontré bien d'autres difficultés qui sont très peu informatiques… Par exemple :

- Il est difficile d'être précis sur le collage et on a vite 2 ou 3 dixièmes de millimètres d'erreur… ce qui a probablement un impact significatif, particulièrement pour les petites tailles.
- L'impression 3D n'est pas parfaite et on a souvent des manques de matière ou au contraire trop de matière par endroit (pas très pratique dans le cadre de ce projet).

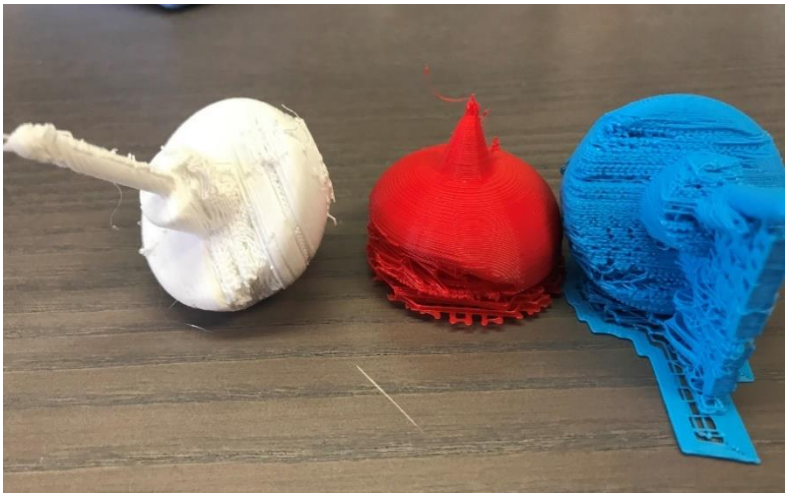
- Parfois, l'impression est même complètement raté ! En fin de semaine dernière, j'avais 80% d'objets qui terminait à la poubelle (la hotline de Dagoma cherche encore la solution)
- … Et je passe sur les problèmes d'encrassement de l'extrudeur, de plantage des capteurs qui amène la buse de l'imprimante à enfoncer son plateau, sur les contraintes exercées sur le plastique (il change de densité en refroidissant) qui peuvent entraîner un décollement de la pièce en cours d'impression (-> poubelle directement), et enfin sur les temps d'impression (typiquement 4h pour imprimer les 5 toupies de taille différentes sur le même plateau).

Impression en cours





Résultat après impression (il faut encore gratter les pièces et coller minutieusement)



Il y a de nombreux échecs avec des manques (ou du trop-plein) de matière

8. Implémenter l'étape d'optimisation sur le modèle simplifié / Vérifier la cohérence

J'ai donc validé qu'en retirant de la matière à des endroits intelligents, on se rapproche d'une toupie optimale. Il s'agit maintenant de décider des creux avec un vrai algorithme d'optimisation et pas juste en bricolant Excel, Blender et Python sur seulement 6 degrés de liberté.

L'article propose de minimiser en fonction de β, où β est le vecteur des densités de chaque feuille non figée de l'octree, de minimiser donc l'énergie suivante :

$$\gamma_{\mathbf{c}} \left(\ell M\right)^2 + \gamma_{\mathrm{I}} \left[\left(\frac{I_a}{I_c}\right)^2 + \left(\frac{I_b}{I_c}\right)^2 \right] + \gamma_{\mathbf{L}} \frac{1}{2} \boldsymbol{\beta}^T \mathbf{L} \boldsymbol{\beta}$$

Sous contrainte (0<= β<=1 ; Sx = Sy = Sxy = Syz =0 ; et la condition (8) de l'article)

Je n'avais aucun background sur la minimisation de ce genre de problème. D'intenses recherches m'ont permis de découvrir qu'il existe de nombreux algorithmes et même des sociétés qui vendent des licences pour utiliser ces algorithmes. Après différents essais sur des cas simples, j'ai souhaité me ramener à un algorithme codé dans Libigl et présenté dans la section Quadratic Programming de la doc en ligne.

Pour cela, il faut formuler le problème sous la forme : $\underset{\mathbf{z}}{\text{minimize}} \ \frac{1}{2}\mathbf{z}^T\mathbf{Q}\mathbf{z} + \mathbf{z}^T\mathbf{B}$. C'est possible en changeant légèrement l'énergie à minimiser :

- Le premier terme $\gamma_{\mathbf{c}} \left(\ell M\right)^2$ peut s'écrire $\beta^T Q \beta$ où Q est la matrice des $\gamma_C z_i z_j$: très bien !
- Le second terme ne va pas ! Mais plutôt que de chercher à minimiser $\left(\frac{I_a}{I_c}\right)^2 + \left(\frac{I_b}{I_c}\right)^2$, on peut minimiser $I_a+I_b-2I_c$ qui a une signification physique claire et s'écrit $\beta^T B$ avec $B_i = \gamma_I (2Z_i^2 – X_i^2 – Y_i^2)$. Au passage, l'angle Φ disparait de l'équation.
- Le troisième terme ne pose pas de problème mais je n'ai pas compris son intérêt tout de suite…

Le problème, c'est que cela ne marche pas ! Et que la documentation de Libigl est plus que sommaire (c'est le problème de l'Open Source…). Bref, déjà, sans le troisième terme (sans le lagrangien), l'appel au solveur renvoi une erreur de calcul.

```
Error: Numerical issue.
Error: min_quad_with_fixed precomputation failed.
```

Après quelques recherches, j'ai compris que l'algorithme ne fonctionnait qu'avec des matrices Q semi définie positives pour avoir des problèmes convexes. C'est ce qu'apporte le lagrangien : même en ajoutant 0.001L, toutes les valeurs propres de Q+0.001L deviennent positives et je n'ai plus d'erreur numérique…

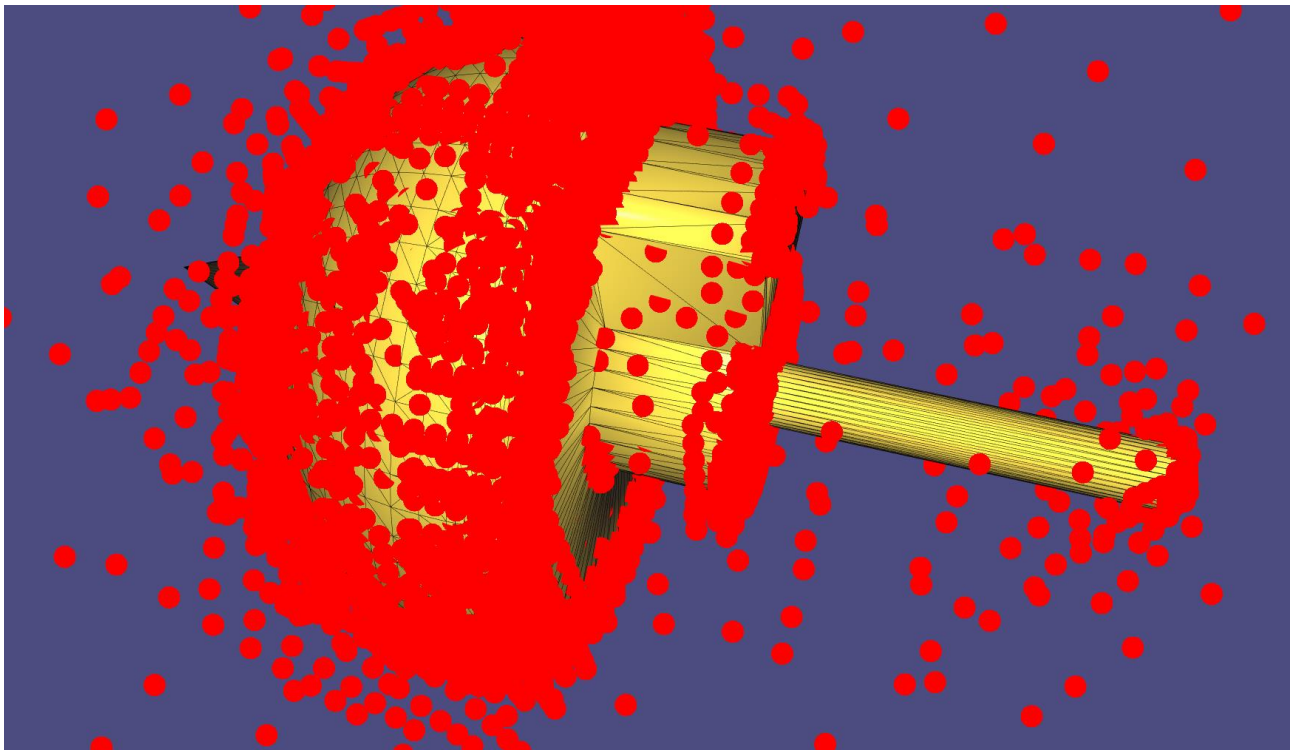
… Mais, alors que cela marchait très bien sur des tests avec de petites matrices 10x10, ma contrainte 0<= β<=1 n'est pas respectée sur une matrice 1098x1098. Pas du tout ! ! ! J'ai des densités à -156 et d'autre à +241. J'ai essayé tous les paramètres du modèle et notamment le nombre d'itérations en espérant une convergence. Mais rien n'y fait. Voici l'erreur cumulée (la somme des écarts à l'intervalle [0,1] sur les 1098 cellules en fonction du nombre d'itérations :

```
Lancement du solveur
ItÚration numÚro 1 faite : erreur cumulee : 137017
ItÚration numÚro 2 faite : erreur cumulee : 22.4901
ItÚration numÚro 3 faite : erreur cumulee : 221105
ItÚration numÚro 4 faite : erreur cumulee : 3.1594e+06
ItÚration numÚro 5 faite : erreur cumulee : 36148.2
ItÚration numÚro 6 faite : erreur cumulee : 27922.5
ItÚration numÚro 7 faite : erreur cumulee : 36531.1
ItÚration numÚro 8 faite : erreur cumulee : 32175
ItÚration numÚro 9 faite : erreur cumulee : 35430.4
```

Il n'y a pas de convergence. Ou plus précisément, l'algorithme semble converger en s'affranchissant d'une contrainte physiquement essentielle… Peut-être est-ce parce que ma matrice Q n'est pas du tout sparse. Peut-être ai-je fait une erreur grossière… mais comme cela marchait pour de petites matrices, je suis tenté de penser qu'il y a un bug sur l'algorithme d'optimisation de Libigl…. Bref, après 15 heures d'essais infructueux, après avoir tenté de passer à MOSEC (un autre algorithme), je me suis résigné à abandonner pour l'instant faute de temps.

9. Implémenter l'algorithme complet (avec split & merge) / Optimiser un objet 3D

Je n'ai donc pas implémenté la fin de l'algorithme. J'ai tout de même fait fonctionné igl::octree pour bien comprendre le résultat et ne vois pas de difficulté particulière à partir de ce code pour l'adapter à mon besoin.



Chaque point rouge est le centre d'une des feuilles de l'octree. Igl::octree raffine jusqu'à n'avoir plus qu'un point de V dans chaque feuille de l'octree.

En guise de conclusion

Je pense avoir compris le cœur de l'article en détail ; j'ai découvert plein de choses en impression 3D et j'ai beaucoup « bricolé » y compris avec Blender que je ne connaissais que de nom ; j'ai réussi à créer des toupies 3D, des vraies, des modifiées, des modifiées creusées. Et mon imprimante 3D a beaucoup de travail devant elle : elle est déjà réquisitionnée par le père Noël pour fabriquer une armée de licornes de toutes les couleurs (j'ai une petite princesse de 4 ans ½)… si la hotline de Dagoma fait des progrès rapidement…

J'aurai buté sur la question de l'optimisation, sans doute faute de bases solides sur le sujet, d'un manque de chance, d'un manque de temps… ou des trois à la fois. En tout état de cause, ce petit voyage dans le monde de l'impression 3D n'est pas fini pour moi, et je compte bien réussir à optimiser mes toupies modifiées en 2020 : sans doute me donnerez vous des pistes pour finaliser mon projet au cours de la soutenance !