Formes paramétriques différentiables

Élie Michel[1] Tamy Boubekeur[2,1]
[1]*LTCI, Télécom Paris, IP Paris* [2]*Adobe*

j.FIG 2020

Résumé

Les formes paramétriques modélisent une géométrie à partir d'un jeu d'hyper-paramètres. Bien qu'elles puissent prendre des aspects variés, leur manipulation se fait en général de façon indirecte et peu intuitive , en manipulant des séries de valeurs plutôt que la géométrie elle-même. Nous proposons une méthodologie basée sur la différentiabilité locale des formes paramétriques usuelles pour offrir une manipulation directe de la géométrie tout en respectant les contraintes de l'espace des paramètres, et ce sans effort supplémentaire lors de la conception de ces formes.

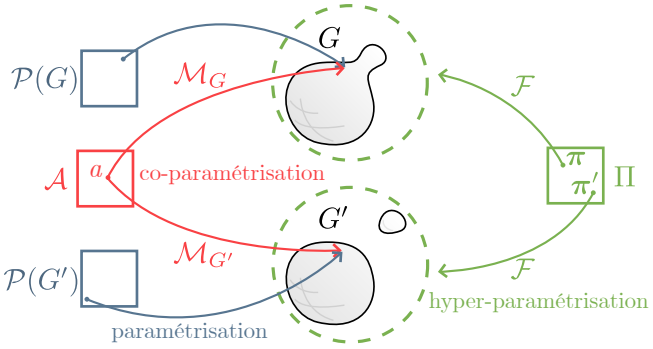

FIGURE 1 – Différentes notions de paramétrisation cohabitent dans cet article. $\mathcal{P}(G) \to G$ est une paramétrisation au sens des « surfaces paramétrées » ; $\mathcal{F} : \Pi \to \{G\}$ est une paramétrisation d'ordre supérieur et $\mathcal{M}_G : A \to G$ est une paramétrisation commune à toutes les géométries issues de $\mathcal{F}$.

1 Introduction

La modélisation *non destructive*, ainsi que le *rigging*, introduisent naturellement des formes paramétriques, dont on peut modifier a posteriori certains hyper-paramètres. On découple ainsi les dimensions techniques et artistiques lors de la création. Cependant, l'interaction avec ces formes passe souvent par une série de valeurs numériques. Nous cherchons ici au contraire à interagir avec les hyper-paramètres directement depuis la vue 3D.

Forme paramétrique $\mathcal{F}$. Une forme paramétrique est représentée par une fonction $\mathcal{F}$ qui à une valuation $\boldsymbol{\pi}$ des hyper-paramètres associe une géométrie statique G :

$$\mathcal{F} : \boldsymbol{\pi} \in \Pi \longmapsto G \subset \mathbb{R}^3 \tag{1}$$

On parle d'*hyper-paramètres* par opposition aux paramètres utilisés pour représenter G sous la forme d'un ensemble paramétré $\mathcal{P}(G) \to G$. Par exemple, si G est un plan fini, son ensemble de paramètres $\mathcal{P}(G)$ peut être $[0,1]^2$. Si c'est une sphère, les paramètres sont canoniquement des coordonnées sphériques, etc.

Hyper-paramètres Π. Les hyper-paramètres sont ceux exposés à l'utilisatrice pour conduire la génération de G. Par exemple, si $\mathcal{F}$ est une forme humanoïde, les hyper-paramètres peuvent être ses mensurations. Plus généralement, l'espace des hyper-paramètres est un compact Π d'un $\mathbb{R}$-espace vectoriel. Ce que l'on appelle *paramètre* Π_k est un vecteur d'une base donnée de Π. Un élément de cet espace est appelé *valuation* et noté :

$$\boldsymbol{\pi} = (\pi_1, \dots, \pi_K) = \sum_{k=1}^{K} \pi_k \Pi_k$$

Cet espace est parfois appelé espace de conception (*design space* [Tal+09]) ou espace de rig [Hah+12] dans des cas spécialisés. Il correspond à l'espace que l'on s'autorise à explorer artistiquement.

Co-paramètres $\mathcal{A}$. Une difficulté majeure pour la manipulation de $\mathcal{F}$ dans son ensemble est que les espaces de paramétrisation $\mathcal{P}(G) = \mathcal{P}(\mathcal{F}(\boldsymbol{\pi}))$ dépendent en général de $\boldsymbol{\pi}$. Par exemple, si les G sont des maillages de triangles, $P(G)$ est usuellement l'ensemble des indices de triangle et coordonnées barycentriques en leur sein, et dépend donc de la connectivité de G.

Nous distinguons alors un espace $\mathcal{A}$ de co-paramétrisation de toutes les géométries qui sont des images de $\mathcal{F}$. Ainsi, on détermine pour tout $G = \mathcal{F}(\boldsymbol{\pi})$ une représentation dont le domaine $\mathcal{A}$ est identique. Les éléments de $\mathcal{A}$ sont appelés *co-paramètres* de $\mathcal{F}$.

Contributions Nous montrons d'une part comment l'existence d'une telle co-paramétrisation permet la création d'une interface de manipulation directe et intuitive d'une forme paramétrique (Section 2). Nous proposons ensuite, dans le cas de maillages, une stratégie d'estimation d'une co-paramétrisation par induction sur le graphe que constitue une implémentation de $\mathcal{F}$ (Section 3).

Notre approche est capable de se greffer à moindres frais à un moteur de forme paramétriques existant, ce que nous montrons dans notre implémentation (Section 4).

Travaux connexes Dans le contexte du *rigging* – l'hyper-paramétrisation d'une géométrie statique existante – la création de contrôleurs dans la vue 3d pour modifier les paramètres de la forme est généralement manuelle. Elle repose sur un assemblage de sous-problèmes de cinématique inverse pour lesquels une solution analytique est dérivable [Dia10]. De nombreux travaux entreprennent de simplifier la déformation de forme [IMH05 ; SA07 ; LG15], parfois basée sur des exemples [Wam16], mais la déformation libre contourne l'hyper-paramétrisation et donc la sémantique de ses contraintes.

Notre approche repose sur une optimisation dans l'espace des hyper-paramètres d'une forme. Un tel processus se retrouve en architecture paramétrique [Yan+11 ; Zha+13], notamment pour trouver des valeurs pour lesquels une forme est constructible [WOD09]. C'est aussi un moyen de faire cohabiter animation manuelle et phénomènes physiques [Hah+12], ou encore de déterminer des hyper-paramètres à partir de photographies de référence [DTM96]. L'exploration de l'espace des paramètres est même envisagé pour la génération procédurale plus lourde [Tal+09 ; Tal+11].

2 Manipulation directe

On suppose dans cette première partie qu'il existe effectivement un ensemble $\mathcal{A}$ de co-paramètres et on cherche à l'exploiter pour permettre une manipulation directe de la forme. La Section 3 montrera comment construire $\mathcal{A}$ dans certains cas pratiques. On note dans ce cas $\mathcal{M}_G$ la bijection qui paramétrise une géométrie statique G par $\mathcal{A}$:

$$\mathcal{M}_G : \mathcal{A} \longrightarrow G$$

Pour plus de commodité, on surcharge la notation $\mathcal{F}$ pour désigner le foncteur $\mathcal{F} : \boldsymbol{\pi} \mapsto \mathcal{M}_G$. Étant donné que $\mathcal{A}$ ne dépend pas de $\boldsymbol{\pi}$, on peut au besoin inverser son ordre de curryfication : on note ainsi $\mathcal{F}_a(\boldsymbol{\pi}) = \mathcal{F}(\boldsymbol{\pi})(a)$ la position du point de co-paramètre $a \in \mathcal{A}$ pour une valuation $\boldsymbol{\pi}$ des hyper-paramètres.

Par *manipulation directe*, on entend le fait que l'utilisatrice exprime son intention dans $\mathbb{R}^3$, au lieu de Π comme c'est le cas avec une interface présentant un curseur par hyper-paramètre. On modélise cette intention par un point source $O \in \mathcal{F}(\boldsymbol{\pi}_0)$ et un point destination $T \in \mathbb{R}^3$. L'objectif de cette section est de déterminer une nouvelle valuation $\boldsymbol{\pi} = \boldsymbol{\pi}_0 + \Delta\boldsymbol{\pi}$ qui approche le « nouveau point O » au plus près de T. Formellement, ce nouveau point est $\mathcal{F}(\boldsymbol{\pi})(a_O)$, où $a_O = \mathcal{F}(\boldsymbol{\pi}_0)^{-1}(O)$ est le co-paramètre de O.

On cherche à atteindre un maximum d'interactivité, ce qui est une difficulté, car $\Delta\boldsymbol{\pi}$ doit être résolu en une fraction de seconde, mais également un avantage, car la mise à jour permanente de $\boldsymbol{\pi}$ permet de supposer petits les $\Delta T = T - O$ et $\Delta\boldsymbol{\pi}$. Considérons alors, en notant $J \in \mathcal{M}_{3,K}$ le jacobien de $\mathcal{F}_{a_O}$, le développement limité au premier ordre :

$$\Delta T \simeq J(\boldsymbol{\pi}_0) \cdot \Delta\boldsymbol{\pi} \tag{2}$$

Deux étapes clefs sont donc nécessaires pour déterminer $\Delta\boldsymbol{\pi}$: évaluer le jacobien J (Section 2.1) et choisir parmi ses pseudo-inverses celle répondant le mieux à l'intuition de l'utilisatrice (Section 2.2).

2.1 Évaluation du jacobien

Trois approches sont envisageables en général : le calcul analytique, l'auto-différentiation lors de l'évaluation de $\mathcal{F}_a(\boldsymbol{\pi})$ ou l'estimation par différences finies. Le calcul analytique, même assisté d'outils de dérivation symbolique, n'est pas toujours envisageable. L'auto-différentiation est une solution efficace, tant que sont prises les précautions pour en limiter la complexité en mémoire. En revanche, y adapter un moteur de forme paramétrique déjà existant peut s'avérer complexe.

Nous adoptons donc les différences finies, qui sont une solution raisonnable car le nombre de paramètres est faible. Cette approche n'introduit pas de surcharge de la mémoire et permet d'utiliser un moteur d'évaluation en boîte noire. Le pas de différentiation – parfois problématique avec cette méthode – est ici suggéré par les bornes imposées aux différents paramètres. Par exemple, si un paramètre Π_k évolue dans l'intervalle $[\alpha_k, \beta_k]$, on utilise un pas $\delta\Pi_k = 10^{-5} \cdot (\alpha_k - \beta_k)$. Les paramètres étant voués à être exposés à l'utilisatrice, on les suppose suffisamment lisses.

Espace écran. Nous avons jusqu'ici supposé que les contraintes étaient données dans $\mathbb{R}^3$. C'est le cas en réalité virtuelle, mais bien souvent les interfaces de manipulation de formes sont projetées dans un espace écran. Dans ce cas, O et T sont des points de l'écran, et on a $Proj : \mathbb{R}^3 \to \mathbb{R}^2$ la projection de la vue de l'utilisatrice. L'Équation 2 devient alors :

$$\Delta T \simeq J_{Proj}(X) \cdot J(\boldsymbol{\pi}_0) \cdot \Delta\boldsymbol{\pi} \tag{3}$$

où $X = \mathcal{F}(\boldsymbol{\pi})(a_O)$ est le point O avant projection. Généralement, cette projection s'exprime sous la forme $Proj(X) = \frac{P \cdot X}{[P \cdot X]_w}$ avec P une matrice de projection arbitraire. Dans ce cas, on en dérive le jacobien suivant :

$$J_{Proj}(X) = \frac{1}{[P \cdot X]_w}\left(P - Proj(X) \cdot P_{w,\cdot}\right) \tag{4}$$

où $P_{w,\cdot}$ est la ligne de P correspondant à la composante w et X est un vecteur colonne. Le reste du raisonnement est inchangé, on inverse $J = J_{Proj}(X) \cdot J(\boldsymbol{\pi}_0)$.

2.2 Inversion du jacobien

La matrice J n'ayant que trois – voire deux – lignes, son rang est faible ; il peut donc y avoir de nombreuses solutions à l'équation 2. Il peut aussi n'y en avoir aucune, auquel cas on considère la projection orthogonale de $\Delta\boldsymbol{\pi}$ sur l'image de J, ce qui minimise l'erreur au sens des moindres carrés. Cette projection, ainsi qu'une base de solutions, est donnée à partir de J^+, la pseudo-inverse de Moore-Penrose de J. Une matrice M est une solution si, et seulement si :

$$\begin{aligned} M &= J^+ + K \cdot \Lambda \\ \text{avec } K &= I - J^+ \cdot {}^tJ \\ \text{et } \Lambda &\text{ une matrice arbitraire} \end{aligned}$$

Régularisation Pour faire le choix d'une solution en particulier, on introduit des objectifs de régularisation visant à correspondre au mieux à l'intuition de l'utilisatrice. Premièrement, le petit mouvement ΔT de l'utilisatrice ne doit pas entraîner une grande variation des paramètres. Autrement dit, la norme L_2 de $\Delta\boldsymbol{\pi} = M\Delta T$ doit être faible. En particulier, si deux hyper-paramètres affectent O selon la même directions, celui des deux ayant le plus d'influence en norme sera privilégié.

Deuxièmement, pour une meilleure lisibilité et anticipation par l'utilisatrice, il est préférable que le changement $\Delta\boldsymbol{\pi}$ ait un maximum de composantes nulles, c'est-à-dire que sa norme L_0 doit être faible.

Heuristiques. Nous avons testé diverses heuristiques de résolution :

Projections Ce solveur naïf projette le déplacement OT sur tous les vecteurs colonne du jacobien pour en déduire les différentes composantes de $\Delta\boldsymbol{\pi}$. Il n'assure que l'énergie $\mathcal{E}$ soit nulle que dans le rare cas où ces colonnes sont deux à deux orthogonales.

Mono-projection Ce solveur détermine à partir du jacobien et de la direction OT un unique paramètre pour lequel effectuer la projection susmentionnée. Il choisi un compromis entre proximité, minimisation de $\mathcal{E}$ et minimisation de la norme L_2 de $\Delta\boldsymbol{\pi}$ en défavorisant les paramètres correspondant à des colonnes de faible norme. Ce solveur inclue un effet d'hystérésis : le paramètre choisi est figé une fois que la distance OT dépasse un seuil de quelques pixels.

Valeurs singulières Ce solveur choisi simplement $M = J^+$. Il assure que $\mathcal{E}$ est minimale, mais tend à affecter tous les paramètres à la fois, ignorant les enjeux de régularisation de $\Delta\boldsymbol{\pi}$.

Valeurs singulières parcimonieuses Ce solveur utilise la solution du solveur précédent pour définir un ordre d'importance des paramètres, puis retire le paramètre de moindre importance et résout à nouveau tant que l'énergie $\mathcal{E}$ résultante n'augmente pas, minimisant la norme L_0 de $\Delta\boldsymbol{\pi}$.

Ces solveurs peuvent être pondérés par des poids assignés aux hyper-paramètres. Ces poids retranscrivent intuitivement l'importance donnée à certains paramètres plutôt qu'à d'autres, et sont essentiels en théorie pour homogénéiser l'espace des hyper-paramètres, dont les axes ont en général des unités très variées.

3 Co-paramétrisation locale

La construction explicite d'une co-paramétrisation d'un continuum de géométries est un problème dont de nombreux travaux se sont emparé [KS04 ; Sch+04 ; BPJ07 ; Bra+08], mais dans notre cas deux remarques permettent de le simplifier.

Premièrement, l'existence d'un co-paramètre unique $a \in \mathcal{A}$ est importante pour le raisonnement, mais la section précédente ne requiert en pratique que l'évaluation de $m_{\boldsymbol{\pi}} = \mathcal{M}_{\mathcal{F}(\boldsymbol{\pi})} \circ \mathcal{M}^{-1}_{\mathcal{F}(\boldsymbol{\pi}_0)}$, la mise en correspondance par la co-paramétrisation de $\mathcal{F}(\boldsymbol{\pi})$ avec $\mathcal{F}(\boldsymbol{\pi}_0)$.

Deuxièmement, une co-paramétrisation locale est suffisante, car $\boldsymbol{\pi}$ est proche de $\boldsymbol{\pi}_0$. On se contente alors de construire une approximation $\overline{m}_{\boldsymbol{\pi}}$ de $m_{\boldsymbol{\pi}}$ qui au lieu de donner un unique point propose un ensemble de candidats $\{X'\}$. On en déduit $m_{\boldsymbol{\pi}}$ en gardant le plus proche :

$$m_{\boldsymbol{\pi}}(X) = \underset{X' \in \overline{m}_{\boldsymbol{\pi}}(X)}{\operatorname{argmin}} \|X - X'\|_2 \tag{5}$$

Induction On modifie la forme paramétrique pour attacher à chacun des points un attribut que l'on nomme *signature* ; $\overline{m}_{\boldsymbol{\pi}}(X)$ est l'ensemble des points de $\mathcal{F}(\boldsymbol{\pi})$ qui ont la même signature que X dans $\mathcal{F}(\boldsymbol{\pi}_0)$.

Lorsqu'une co-paramétrisation est connue[1], elle est utilisée comme signature. On suppose maintenant que $\mathcal{F}$ est de la forme $\boldsymbol{\pi} \mapsto f_{\boldsymbol{\pi}}\left(\mathcal{F}^{(1)}(\boldsymbol{\pi}), \ldots, \mathcal{F}^{(n)}(\boldsymbol{\pi})\right)$, combinaison par un opérateur $f_{\boldsymbol{\pi}}$ d'autres formes paramétriques $\mathcal{F}^{(i)}$ qui par hypothèse d'induction émettent une signature pour chaque point.

On exploite alors la capacité supposée de l'opérateur $f_{\boldsymbol{\pi}}$ à transférer de façon cohérente les attributs arbitraires des points d'entrée à ceux de sortie. Cette hypothèse sur $f_{\boldsymbol{\pi}}$ est raisonnable car souvent déjà vérifiée par les outils de modélisation non destructive existants. On ajoute également l'indice i à la signature de tous les points de $\mathcal{F}^{(i)}(\boldsymbol{\pi})$ en amont de $f_{\boldsymbol{\pi}}$. Plusieurs points de sortie peuvent se voir transférer la même signature, expliquant que $\overline{m}_{\boldsymbol{\pi}}(X)$ est un ensemble et non un unique point.

1. Par exemple en cas de maillages de connectivité constante, ou lorsque les UV forment une bonne paramétrisation.

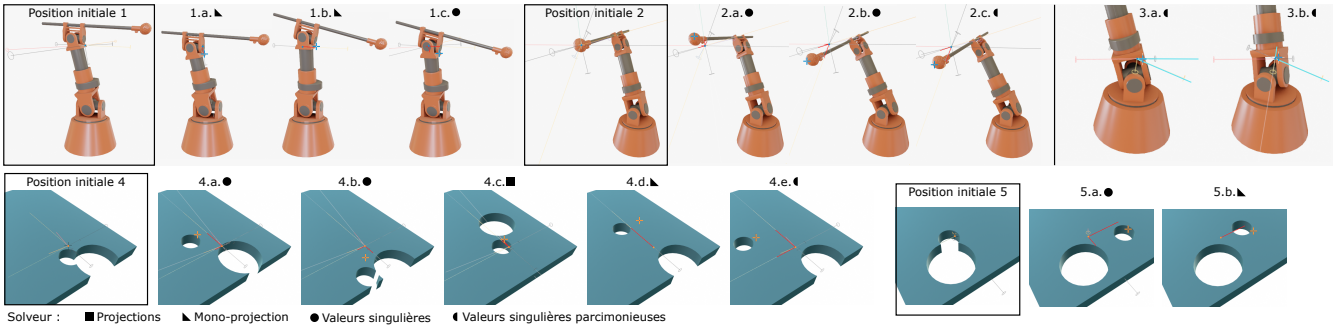

FIGURE 2 – Exemples de résolutions du changement d'hyper-paramètre sur différentes scènes et avec différents solveurs. La seconde scène présente une connectivité variable. Le point bleu/orange est l'origine O du déplacement et la croix est la position actuelle T de la souris. Les lignes rouges représentent la différence appliquée à chaque paramètre par le solveur. L'exemple 3 montre un défaut du solveur *Valeurs singulières parcimonieuses* : son choix de paramètres change brutalement entre deux positions de T proches.

On transforme ainsi la capacité des opérations $f_{\boldsymbol{\pi}}$ qui constituent l'implémentation de la forme paramétrique $\mathcal{F}$ à mettre en correspondance leur entrée et leur sortie en une correspondance (locale) entre plusieurs sorties.

4 Implémentation

La Figure 2 montre l'interface (*add-on* Blender) exposée à l'utilisatrice : il suffit de cliquer et tirer sur n'importe quel point de la surface pour modifier les hyper-paramètres du modèle, optimisés par le solveur selectionné (Section 2.2). Toutes ces opérations sont guidées par une prévisualisation du jacobien.

Tampon par sommet. Notre implémentation précalcule le jacobien de $\mathcal{F}_a$ pour les co-paramètres de tous les sommets du maillage et les y sauvegarde. Ce tampon ne doit être mis à jour que lorsque la valuation change, mais est indépendant du point de vue de l'utilisatrice. Un même échantillonnage de $\mathcal{F}(\boldsymbol{\pi})$ peut donc être utilisé pour estimer tous les $m_{\boldsymbol{\pi}}(X)$; on met ainsi en commun la construction d'une structure d'accélération pour efficacement évaluer l'argmin de l'Equation 5. Nous nous sommes contenté de tables de hachage pour de petits maillages, mais il serait judicieux d'utiliser un arbre k-d (sur l'espace de co-paramètres $\mathcal{A}$ et non sur $\mathbb{R}^3$).

Indicateur visuel. La mise en tampon des jacobiens permet de plus d'afficher un indicateur au simple survol par le curseur de l'utilisatrice, avant même le début de l'interaction. On dessine les vecteurs colonnes du jacobien de $Proj \circ \mathcal{F}_a$ (voir Figure 2). Chacun de ces vecteurs correspond à un hyper-paramètre (identifié dans notre implémentation par un code couleur), et est un vecteur de l'espace de manipulation. Il indique l'influence locale de l'hyper-paramètre sur la position du point survolé, et permet donc à l'utilisatrice d'anticiper.

Tampon de jacobiens. Notre implémentation propose une variante dans laquelle le jacobien est moyenné sur un voisinage du curseur. Dans un tel cas, l'ensemble des jacobiens $J_{Proj \circ \mathcal{F}_a}$ de l'écran est précalculée à chaque changement de point de vue. On effectue pour cela une rastérisation du maillage étiqueté aux sommets par $J_{\mathcal{F}_a}$, ce qui permet d'utiliser l'accélération matérielle (GPU). La taille de la brosse est fixe en espace écran, ce qui permet de donner une influence au niveau de zoom de la vue sur la solution du solveur.

5 Discussion

En l'état actuel, notre prototype permet déjà une interaction instinctive avec une forme paramétrique directement dans la vue 3D pour les cas où une construction claire de la mise en correspondance $m_{\boldsymbol{\pi}}$ est possible. Il reste cependant beaucoup de choses à explorer, à la fois du côté de cette mise en correspondance et dans l'espace des inverses possibles du jacobien.

Mise en correspondance Notre construction d'une correspondance $m_{\boldsymbol{\pi}}$ entre plusieurs sorties différentes pourrait par exemple s'appuyer sur des vecteurs d'attributs aléatoires aux sommets d'entrée et sur une meilleure structure d'accélération pour résoudre l'Equation 5.

Solveurs D'autres types de régularisation de $\Delta\boldsymbol{\pi}$ sont envisageables pour le choix de l'inverse du jacobien. Nous envisageons en particulier des critères de conservation globaux, comme par exemple le volume. D'autre part, nous figeons actuellement $\boldsymbol{\pi}_0$ pour tout un tracé de l'utilisatrice, mais on pourrait réévaluer J de temps en temps – toutefois pas aussi fréquemment que $\Delta\boldsymbol{\pi}$ car ce serait coûteux. Cela donnerait une information d'ordre supérieur au premier degré dans la direction du déplacement et limiterait effectivement la distance entre $\boldsymbol{\pi}_0$ et $\boldsymbol{\pi}$.

Références

[BPJ07] Janine Bennett, Valerio Pascucci et Kenneth Joy. « Genus Oblivious Cross Parameterization : Robust Topological Management of Inter-Surface Maps ». In : oct. 2007, p. 238-247. doi : 10.1109/PG.2007.42.

[Bra+08] Derek Bradley et al. « Markerless Garment Capture ». In : *ACM Transactions on Graphics (Proc. SIGGRAPH 2008)* 27.3 (2008), p. 99.

[Dia10] Rosen Diankov. « Automated Construction of Robotic Manipulation Programs ». Thèse de doct. Carnegie Mellon University, Robotics Institute, août 2010. url : http://www.programmingvision.com/rosen_diankov_thesis.pdf.

[DTM96] Paul D. Debevec, Camillo J. Taylor et Jitendra Malik. « Modeling and Rendering Architecture from Photographs : A Hybrid Geometry-and Image-Based Approach ». In : *Proceedings of the 23th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '96. 1996.

[Hah+12] Fabian Hahn et al. « Rig-Space Physics ». In : *ACM Trans. Graph.* 31.4 (juill. 2012). issn : 0730-0301. doi : 10.1145/2185520.2185568. url : https://doi.org/10.1145/2185520.2185568.

[IMH05] Takeo Igarashi, Tomer Moscovich et John F. Hughes. « As-Rigid-as-Possible Shape Manipulation ». In : *ACM Trans. Graph.* 24.3 (juill. 2005), p. 1134-1141. issn : 0730-0301. doi : 10.1145/1073204.1073323. url : https://doi.org/10.1145/1073204.1073323.

[KS04] Vladislav Kraevoy et Alla Sheffer. « Cross-Parameterization and Compatible Remeshing of 3D Models ». In : *ACM Trans. Graph.* 23.3 (août 2004), p. 861-869. issn : 0730-0301. doi : 10.1145/1015706.1015811. url : https://doi.org/10.1145/1015706.1015811.

[LG15] Zohar Levi et Craig Gotsman. « Smooth Rotation Enhanced As-Rigid-As-Possible Mesh Animation ». In : *IEEE Transactions on Visualization and Computer Graphics* 21.2 (fév. 2015), p. 264-277. issn : 1941-0506. doi : 10.1109/TVCG.2014.2359463.

[SA07] Olga Sorkine et Marc Alexa. « As-Rigid-as-Possible Surface Modeling ». In : *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*. SGP '07. Barcelona, Spain : Eurographics Association, 2007, p. 109-116. isbn : 9783905673463.

[Sch+04] John Schreiner et al. « Inter-Surface Mapping ». In : *ACM SIGGRAPH 2004 Papers*. SIGGRAPH '04. Los Angeles, California : Association for Computing Machinery, 2004, p. 870-877. isbn : 9781450378239. doi : 10.1145/1186562.1015812. url : https://doi.org/10.1145/1186562.1015812.

[Tal+09] Jerry O. Talton et al. « Exploratory Modeling with Collaborative Design Spaces ». In : 28.5 (déc. 2009), p. 1-10. issn : 0730-0301. doi : 10.1145/1618452.1618513. url : https://doi.org/10.1145/1618452.1618513.

[Tal+11] Jerry O. Talton et al. « Metropolis Procedural Modeling ». In : *ACM Trans. Graph.* 30.2 (avr. 2011), 11 :1-11 :14. issn : 0730-0301. doi : 10.1145/1944846.1944851. url : http://doi.acm.org/10.1145/1944846.1944851.

[Wam16] Kevin Wampler. « Fast and Reliable Example-Based Mesh IK for Stylized Deformations ». In : *ACM Trans. Graph.* 35.6 (nov. 2016). issn : 0730-0301. doi : 10.1145/2980179.2982433. url : https://doi.org/10.1145/2980179.2982433.

[WOD09] Emily Whiting, John Ochsendorf et Frédo Durand. « Procedural modeling of structurally-sound masonry buildings ». In : (2009).

[Yan+11] Yong-Liang Yang et al. « Shape Space Exploration of Constrained Meshes ». In : *Proceedings of the 2011 SIGGRAPH Asia Conference*. SA '11. Hong Kong, China : Association for Computing Machinery, 2011. isbn : 9781450308076. doi : 10.1145/2024156.2024158. url : https://doi.org/10.1145/2024156.2024158.

[Zha+13] Xin Zhao et al. « Intuitive Design Exploration of Constrained Meshes ». In : *Advances in Architectural Geometry 2012*. Sous la dir. de Lars Hesselgren et al. Vienna : Springer Vienna, 2013, p. 305-318. isbn : 978-3-7091-1251-9.

Annexes

A. Performances

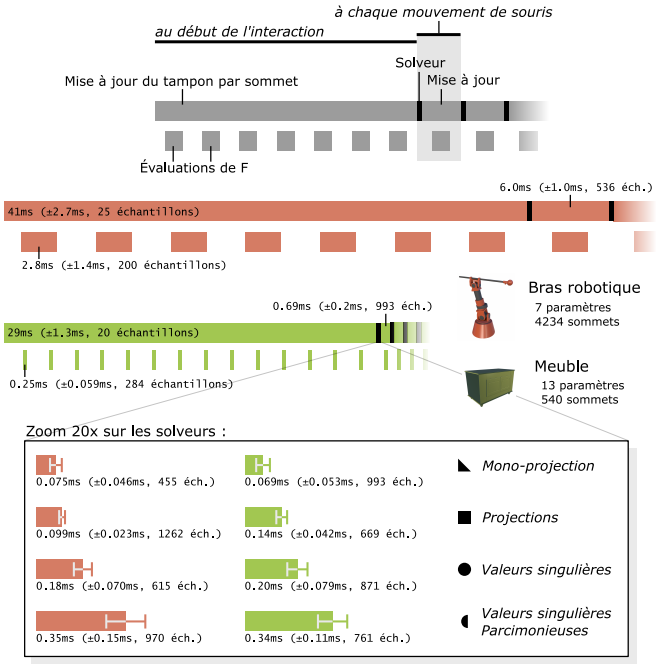

Figure 3 – Résultats de mesures de performance des différentes étapes de notre méthode.

La Figure 3 donne des mesures de temps d'exécution sur deux scènes. L'évaluation des différences finies prend une grande partie du temps mais n'est pas exécutée aussi souvent que le solveur. Le temps d'exécution du solveur reste inférieur au temps d'application des changements (réévaluation de la forme).