Global Parametrization Algorithms for Quadmeshing

Guillaume Coiffier

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Global Parametrization Algorithms for Quadmeshing

THÈSE

présentée et soutenue publiquement le 6 décembre 2023
pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Guillaume Coiffier

réalisée sous la direction de

Dmitry Sokolov et Etienne Corman

Composition du jury

Président :	**Xavier Goaoc**	Université de Lorraine - LORIA
Rapporteurs :	**Mirela Ben-Chen**	Technion, Institut de Technologie d'Israel
	Marco Tarini	Université de Milan
Examinateurs :	**Pooran Memari**	CNRS - Ecole Polytechnique
	Mélina Skouras	Centre Inria de l'Université de Grenoble
	Bruno Lévy	Centre Inria de l'Université de Lorraine
	Etienne Corman	CNRS - LORIA
Invité :	**Dmitry Sokolov**	Université de Lorraine - LORIA

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Résumé

Les maillages quadrangulaires (quads) sont une structure de données centrale au domaine du traitement automatique de la géométrie, trouvant des applications en infographie comme en simulation numérique. Une approche prometteuse pour générer automatiquement des maillages quads de grande qualité s'appuie sur le fait qu'ils constituent une déformation de la grille régulière presque partout, excepté en un petit nombre de points singuliers. Grâce au calcul d'une paramétrisation, à savoir une représentation planaire, de la surface à mailler, il est alors possible d'y tracer une grille qui, reprojetée sur la surface, formera le maillage désiré. Pour que des quadrilatères puissent être extraits, cette paramétrisation se doit d'être « sans couture », c'est-à-dire de respecter un ensemble de contraintes d'alignement sur son bord et ses découpes. Ces contraintes sont généralement imposées petit à petit dans un pipeline d'opérations désormais bien étudié, consistant en un calcul de champ de repères lisse, définissant les futurs points singuliers du maillage, une phase d'intégration pour obtenir une paramétrisation aux coutures sans rotation, suivie d'une phase de quantification déterminant les degrés de liberté en translation.

Cette thèse s'intéresse à l'amélioration des différentes étapes du pipeline de génération de maillages quadrangulaires. En nous appuyant sur des notions de géométrie différentielle, nous proposons des formulations du problème évitant les écueils de l'approche actuelle.

Premièrement, nous abandonnons la résolution de problèmes en nombre entier pour certaines étapes (connue pour être difficiles à résoudre) pour la remplacer par la minimisation de fonctions objectif continues (bien que non convexe).

Deuxièmement, nous fusionnons certaines étapes du pipeline en une seule optimisation déterminant en un seul coup les degrés de liberté correspondants. Cela permet plus de versatilité et de contrôle utilisateur sur le maillage quad final, et évite les cas d'échecs classiques causés par l'approche gloutonne du pipeline actuel. Ces formulations théoriques du problème de paramétrisation sans couture s'accompagnent d'implémentations pratiques dans lesquelles nous démontrons la viabilité de nos approches sur une grande variété de modèles CAO.

Finalement, notre travail est en théorie généralisable au problème plus difficile du maillage hexaédrique, là où les algorithmes de paramétrisation actuels sont soit uniquement valables pour les surfaces, soit échouent à produire des résultats de façon robuste.

Mots-clés: Maillage quadrangulaire, Paramétrisation globale, Géométrie numérique

Abstract

Quadrangular meshes are a central data structure in the domain of geometry processing, with applications ranging from computer graphics to numerical simulation. A promising approach for automatic generation of high quality quadmeshes relies on the observation that they consist in a deformation of the regular grid almost everywhere, except at a few singular vertices. Thanks to the computation of a parametrization, that is to say a flat representation, of the surface to be meshed, it is then possible to overlay a grid to be lifted back as a quad mesh. For quads to be extracted from this surface parametrization, it has to be seamless, that is to say satisfy a set of alignment constraints on its boundary and its cuts. This needs a quantization as integer values of some of its degrees of freedom. These constraints are usually enforced progressively in a well-studied pipeline of operations, consisting in the computation of a smooth frame field, defining the future singular vertices of the mesh, an integration phase from which a rotationally seamless parametrization is recovered, and a quantization step to determine the translational degrees of freedom.

In this work, we focus on improving the steps of the parametrization-based quadmeshing pipeline. Using notions from differential geometry, we propose formulations of the problem that avoids the drawbacks of the current approaches.

Firstly, we get rid of the mixed-integer optimization problems (known to be hard to solve) used in some steps. We replace them by the minimization of smooth (yet non-convex) objective functions.

Secondly, we merge some of the pipeline's steps into a single optimization determining the corresponding degrees of freedom in one go. This allows for more versatility and user control over the final quadmesh, and avoids typical failure cases caused by the greedy approach of the current pipeline. These theoretical formulations of the seamless parametrization problems are accompanied by practical implementations where we demonstrate the viability of our approach on a vast array of CAD models.

Finally, our work is theoretically generalizable to the more difficult problem of hexahedral meshing. As current parametrization-based approaches are only suitable for surfaces or simply fail at reliably producing results in the volume case, this opens the way for more robustness and qualitative hexmeshes.

Keywords: Quadmesh, Global parametrization, Geometry processing

Remerciements

Tout d'abord, je tiens à remercier les membres de mon jury de thèse : Mirela Ben-Chen, Marco Tarini, Pooran Memari, Mélina Skouras, Bruno Lévy et Xavier Goaoc. Je remercie tout particulièrement Mirela Ben-Chen et Marco Tarini pour avoir lu et rapporté mon manuscrit, et dont les précieux retours n'auront pas manquer de l'améliorer.

Cette thèse n'aurait évidemment pas pu voir le jour sans mes directeurs de thèse. Merci à Dmitry pour ses précieux conseils, son soutien et sa confiance. Merci à Étienne pour son amour de la géométrie, son investissement dans nos projets et sa patience pour m'avoir supporté en tant que co-bureau pendant près de trois ans et demi.

Je salue également les autres membres de l'équipe PIXEL, passés et présents, avec qui j'ai eu le plaisir de travailler, discuter, plaisanter, et manger du fromage : Nicolas, Laurent, Bruno, Dobrina, François, Justine, Yoann, David et David, sans oublier nos partenaires industriels privilégiés Cédric et Alex.

Cette thèse a aussi été l'occasion pour moi de m'investir pleinement dans la médiation scientifique. J'ai pris énormément de plaisir à parler de cartes, de peau de clémentine, de lapins aplatis et de bien d'autres choses aux jeunes et aux moins jeunes. Je remercie le public qui a pris le temps de m'écouter parler de ma thèse mais également les organisateurs des différents événements auquel j'ai participé, notamment Véronique Poirel, pour sa gestion du programme *Chiche!*, Étienne Haouy et Catherine Flauder.

Je remercie chaleureusement ma famille pour leur amour et leur soutien inconditionnels depuis plus de 28 ans maintenant : ma mère Lydie, mon père Jean-Christophe, mon petit-frère Thibault, mon oncle et parain Pascal, mon grand-père Roger.

Mes pensées vont également à mes deux grands-mères, qui nous ont malheureusement quittés au tout début de cette thèse : Jeannine, pour qui les études étaient si importantes, et Gilberte, qui n'aurait pas manqué de glisser un compliment à mon égart à toutes les personnes présentes à la soutenance.

Nombreuses sont les personnes avec qui je me suis lié d'amitié au fil des années et à qui je voudrais rendre hommage ici, au souvenir des bons moments passés ensemble.

Celles et ceux que j'ai rencontré à Nancy, qui y ont habité en même temps que moi ou qui ont eu le courage de venir m'y rendre visite : Juliette, Léo, Bastien, Quentin, Raphael, Ariane, Valentin, Mélora, Florimond. Je salue les membres de notre petit (mais grandissant) groupe d'escalade bi-hebdomadaire : Noémie, Floriane, Louis, Julia[1].

Mes anciens colocataires lyonnais du 1 Avenue Debourg : Louis[2], Joël, Nicolas et Danaé.

Merci au *4e Étage* pour les grandes bouffées d'air frais que constituaient ses jeux, ainsi qu'à tous ses membres avec qui j'ai vécu les plus grandes aventures imaginaires ainsi que beaux moments de vie réelle : Adrien, Georges, Janelle, Malo, Marie, Alix, Corentin, Mista, Juliette, Xavier, Laureline, Bertrand, Morgan, Florine, Lambert, Nattes, Aaren, Lison, Simon, Gabrielle, Antoine, Bastien, Youssed, Simon, Colin, Marine, Elodie, Olivier, Marlysa, Dana, je vais forcément en oublier parmis vous et je m'en excuse. Vous êtes beaucoup, mais sentez vous remerciés.

[1]Mais il faut quand même se rendre à l'évidence : l'escalade, je ne suis pas vraiment fait pour ça.

[2]La thèse nous a enfin permis d'être co-auteurs, depuis le temps qu'on en parlait

Un grand merci au serveur Discord des patates, fidèle public du flot quotidien de memes, rendus blender de lapin en fromage ou encore captures d'écrans de bugs graphiques comiques en tout genre. Garder ce contact virtuel avec vous malgré la distance fut tout aussi important pour moi qu'il fut dommageable à ma productivité au bureau.

J'en profite pour glisser un remerciement à Keenan Crane, créateur des modèles 3D de *Spot* la vache et *Bob* la bouée canard, ce dernier ayant constitué la figure *placeholder* du manuscrit en attendant que les autres soient terminées (voir Figure 0), et ainsi contribué grandement à la conservation de ma santé mentale durant la rédaction de ce manuscrit.

Pour finir, j'aimerais remercier ma plus fidèle collocataire pendant ces années de thèse : Maple la lapine[3]. Sans oublier son inénarrable maitresse, Justine, qui partage et illumine ma vie depuis près de trois ans maintenant. Merci pour ta présence à mes côtés, ton soutien et ton affection dans les bons comme les mauvais moments[4].

Figure 0: Last occurence of the *Placeholder* image used during the redaction of this manuscript

[3] qui n'a pas été aplatie, elle.

[4] et merci aussi pour la figure 2.2b

Moi, dans la vie, j'aplatis des lapins...

Contents

Chapter 1 Introduction (Français) **1**

Chapter 1 Introduction (English) **11**

Chapter 2 Background and State of the Art in Surface Parametrization **19**

2.1 Elements of Differential Geometry 20
2.2 Surface Meshes 29
2.3 Surface Parametrization for Texture Mapping 37
2.4 Seamless and Grid-preserving Parametrization Applied to Quadmeshing 50

Chapter 3 Computing Cone Distributions via Continuous Optimization on a 4-covering **69**

3.1 4-covering of a Triangle Mesh 71
3.2 Flattening Energy 73
3.3 Optimization of the Flattening Energy 77
3.4 Results and Discussion 82
3.5 Conclusion 82

Chapter 4 The Method of Moving Frames for Surface Global Parametrization **85**

4.1 Related Work 88
4.2 Smooth Formulation 89
4.3 Discrete Setting 93
4.4 Local Frames 94
4.5 First Structure Equation 96
4.6 Quantized Cone Parametrization 98
4.7 Boundary and Feature Edge Adaptation 102
4.8 Numerical Optimization 103
4.9 Applications and Evaluation 109
4.10 Conclusion and Perspectives 114

Chapter 5 Alternative Discretization for the Method of Moving Frames **119**
5.1 Face-based Discretization 120
5.2 Numerical Optimization 124
5.3 Results and Comparison 128
5.4 Conclusion and Perspectives 131

Chapter 6 Grid-preserving Parametrization in a Single Optimization **137**
6.1 Periodic Global Parametrization 138
6.2 Motivations 143
6.3 Seam-agnostic Oscillator Functions 144
6.4 Parametrization Setup 150
6.5 Conclusion 153

Chapter 7 Conclusion and Perspectives **155**
7.1 Summary of Contributions 155
7.2 Future Works in Quadmeshing 156
7.3 Towards Parametrization Methods for Hexmeshing 158

Bibliography **161**

1

Introduction

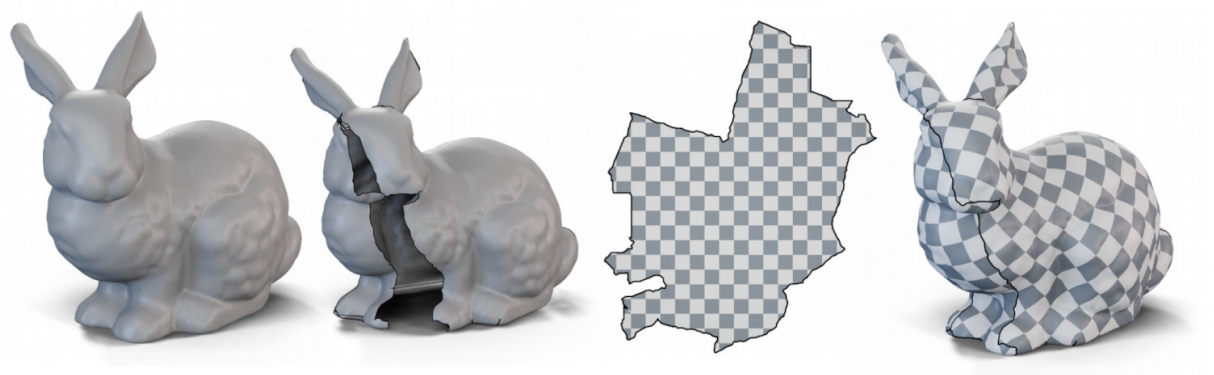


Figure 1.1: Paramétrisation globale du *Lapin de Stanford*. Une surface, représentée par un maillage triangulé (gauche) est découpée selon d'un ensemble de coutures (milieu gauche) et aplatie vers le plan avec peu de distortion (milieu droite). Une grille régulière, représentée par une texture en damier, peut être superposée à cette représentation plate puis plaquée sur la surface initiale (droite).

Cartes, Coutures et Courbure

La géométrie est l'étude des formes. Son formalisme mathématique nous permet de comprendre les objets qui nous entourent, depuis les effets de la relativité générale à l'échelle cosmique jusqu'aux objets les plus banals comme les vêtements cousus ou les emballages de bonbons, ce qui a permis de nombreuses percées dans les domaines de la science et de la technologie.

Comment décrire mathématiquement la forme d'un objet ? Imaginons par exemple une surface lisse dans un espace tridimensionnel. Une possibilité pour la décrire complètement serait d'énumérer l'ensemble de tous les points de l'espace qui lui appartiennent. Bien qu'il s'agisse d'une description valable de notre surface, procéder ainsi ne nous donne aucune information concernant sa structure. En effet, une surface est un objet bidimensionnel, ce qui signifie que la trajectoire d'un point s'y déplaçant est limitée à deux degrés de liberté : "avant/arrière" et "gauche/droite". De ce point de vue local, une surface se comporte comme un plan euclidien classique. Cela signifie qu'il est possible d'indiquer précisément la position d'un point en indiquant la distance à parcourir "vers la gauche" et "vers l'avant" pour atteindre ce point depuis

une origine arbitraire. En d'autres termes, cela revient à doter notre surface d'un système de coordonnées bidimensionnel cohérent, ce qui revient à la déplier et à la considérer comme un objet plat. Ce processus est appelé *paramétrisation* (ou paramétrage) de la surface.

Cette idée de représentation d'une surface dans le plan intervient dans de nombreuses applications de la vie quotidienne. La première et peut-être la plus évidente d'entre elles est la fabrication de cartes géographiques, qui ne sont ni plus ni moins qu'une paramétrisation de la totalité, ou d'une partie, de la surface de la Terre. Notre planète était généralement identifée à une sphère parfaite, le calcul de telles cartes peut se faire analytiquement ou à la main, et intègre généralement plus de contraintes géopolitiques que de contraintes mathématiques.

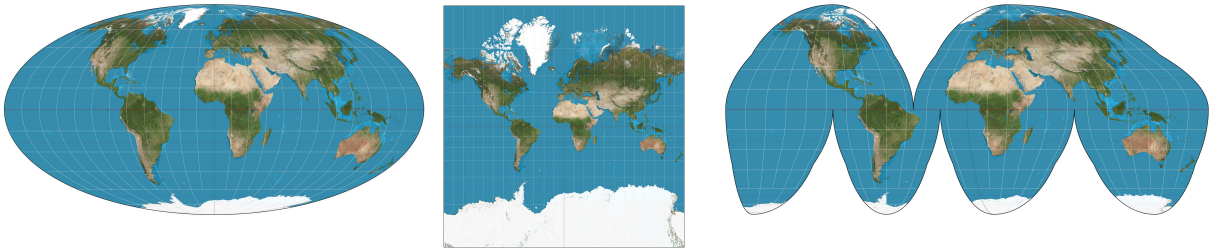
Une autre tâche étroitement liée à la paramétrisation de surface est la fabrication de vêtements, où des pièces de tissu initialement plates doivent être cousées pour former un objet non plat. Là encore, comme les vêtements présentent une forme globale commune, la décomposition en motifs de tissus (devant, dos, manches, etc.) a été réalisée sans ordinateur pendant des siècles, bien que l'automatisation de la fabrication et de la conception des vêtements soit un domaine de recherche actif [Nayak and Padhye 2017].

Ce n'est que plus récemment, depuis une cinquantaine d'années, que l'apparition de l'informatique graphique 3D a fait naître le besoin d'algorithmes automatiques pour la paramétrisation de surfaces. Dans les jeux vidéos et les films d'animation, le calcul et le stockage d'une représentation plane d'un objet donné fait en effet partie de la technique canonique pour lui donner des couleurs et des détails. En raison de la variété et du nombre d'objets à traiter dans une seule scène, l'automatisation de l'étape de paramétrisation représente donc un gain de temps et de qualité essentiel.

Dans toutes ces applications, il n'est pas seulement nécessaire de calculer une carte d'un objet, mais aussi de respecter certaines contraintes. Pour être utiles, les cartes doivent en effet présenter certaines propriétés naturelles. La première est que la surface doit être représentée aussi précisément que possible. En d'autres termes, une forme donnée dessinée sur la surface ne doit pas être déformée lorsqu'elle est aplatie, tant en termes de taille (mise à l'échelle) que de déformation (cisaillement, étirement). Dans le cas des cartes géographiques, on s'attend en effet à ce que les masses continentales et les océans aient des formes reconnaissables et que leurs tailles relatives restent constantes. Malheureusement, cela s'avère impossible : une carte "isométrique" de la Terre ne peut pas exister mathématiquement. Il faut trouver un compromis entre la préservation parfaite des aires relatives (comme la projection de Mollweide de la Figure 1.2a) au détriment de l'étirement, ou la préservation parfaite des formes (comme la projection de Mercator, Figure 1.2b) au détriment de changements parfois spectaculaires dans les surfaces relatives. En 2023, la page Wikipedia listant les types de projection pour les cartes de la Terre[5] contient plus de 80 entrées, illustrant le besoin de cartes différentes en fonction de l'application ciblée.

Une autre propriété importante des paramétrisations de surface est l'injectivité : chaque point de la carte doit correspondre au maximum à un point de la surface. En effet, une carte qui place Paris et Tokyo au même endroit risque d'être impossible à lire. Cette contrainte a une conséquence importante : selon la topologie de la surface considérée, il peut être nécessaire d'effectuer des découpes pour pouvoir déplier correctement la surface. Dans le cas d'une sphère comme la Terre, il est habituel d'effectuer une coupe du pôle nord au pôle sud en passant par l'océan. De même, un vêtement comme un T-shirt ne peut être fabriqué qu'à partir d'une pièce de tissu dont les manches ont été découpées en rectangles avant d'être cousues en forme cylindrique. C'est de ce dernier exemple que vient la dénomination courante des coupes dans

[5] `https://en.wikipedia.org/wiki/List_of_map_projections`



(a) Mollweide (b) Mercator (c) Goode homolosine

Figure 1.2: Différentes projections de la surface de la Terre vers le plan. (a) La projection de Mollweide conserve les aires et les tailles relatives des continents. (b) La projection de Mercator est *conforme* et préserve les angles et les formes au détriment d'importantes variations de taille au niveau des poles. Même s'il est mathématiquement impossible de calculer une carte préservant à la fois les aires et les angles, l'optimum peut être approché en introduisant des découpes (c). Images extraites de *Wikipedia*[5].

une paramétrisation de surface : les coutures.

Les coutures d'une paramétrisation, nécessaires pour des raisons topologiques, peuvent également être ajoutées pour minimiser la distorsion. Cette idée vient d'une intuition physique : lorsque nous considérons l'aplatissement d'un objet réel, comme la peau d'une clémentine (Figure 1.3), la rigidité du matériau empêche les grandes déformations. Au lieu de cela, l'objet est plus susceptible de se déchirer lorsqu'il est aplati, créant ainsi une couture, et la paramétrisation résultante est en moyenne moins (comme avec la projection de Goode de la Figure 1.2c). En plus du compromis entre les distorsion d'aires et d'angles vient donc s'en ajouter un autre, entre la quantité de discontinuités et la distorsion de la carte.

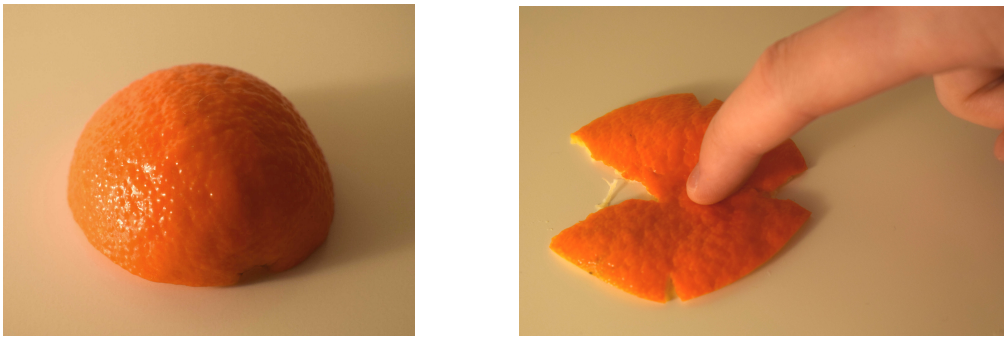


Figure 1.3: Dans le monde réel, tenter d'aplatir un objet a généralement pour conséquence l'apparition de déchirures, comme c'est le cas pour cette peau de clémentine. Les déchirures apparaissent naturellement comme un moyen de soulager la distorsion du matériau.

Étant donné une surface lisse arbitraire, le calcul d'une paramétrisation de bonne qualité nécessite donc de répondre à un certain nombre de questions : Doit-on plutôt préserver les aires ou les angles ? Est-il préférable d'avoir une carte rectangulaire connectée avec une forte distorsion, ou une carte à faible distorsion avec de nombreuses coutures ? Étant donné une surface, quelle est sa meilleure paramétrisation possible sous un certain critère de distorsion ou de forme du bord ? Au delà de ces questions générales, certaines applications, comme celle qui nous intéresse dans le cadre de ce travail, exigent des contraintes supplémentaires sur la cartographie finale, telles que l'alignement vertical ou horizontal des bords de la carte ou une distorsion contrôlée

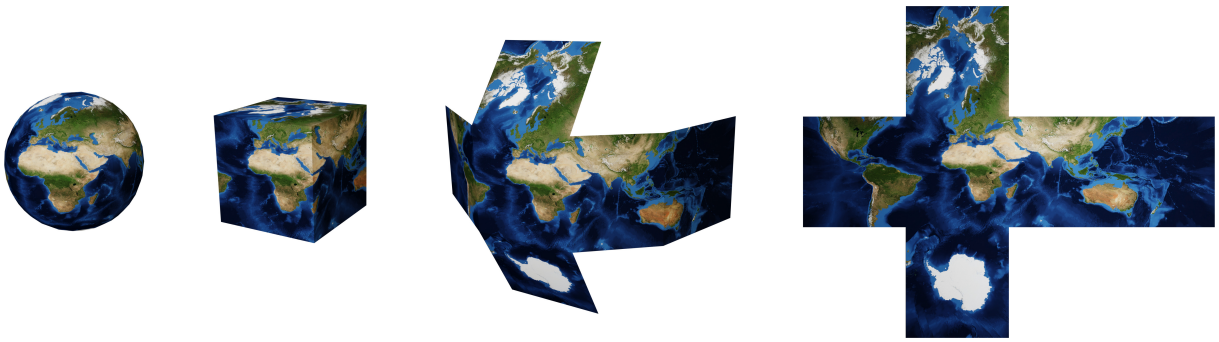


Figure 1.4: Une paramétrisation globale fait correspondre la distribution de la courbure d'une surface lisse à un ensemble fini et réduis de cônes. Ici, 8 cônes concentrent chacun $\pi/2$ de courbure, ce qui revient à transformer la Terre en un cube. La paramétrisation est ensuite obtenue en coupant le cube entre les cônes et en dépliant le patron sans distorsion supplémentaire.

au travers des coutures, ce qui rend le problème de la recherche d'une paramétrisation optimale encore plus difficile.

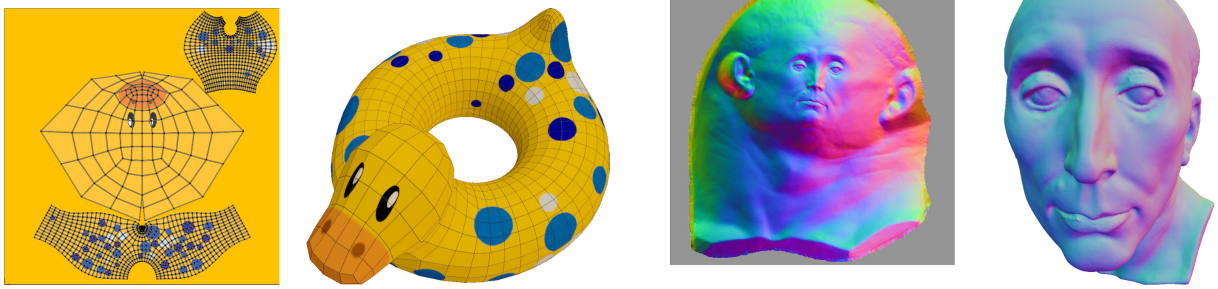
Le concept de paramétrisation *globale* constitue une avancée prometteuse vers la réponse à ces questions. Il repose sur l'observation clé selon laquelle la distorsion est causée par un changement de courbure. La courbure, et plus précisément la courbure gaussienne, est un nombre qui quantifie la manière dont une surface diffère localement d'une surface plane. Étant donné que la courbure peut être distribuée arbitrairement sur une surface mais qu'elle n'est présente qu'aux limites de sa représentation plane, tout calcule d'une paramétrisation doit la déplacer d'une façon ou d'une autre, et ce faisant, inévitablement créer des distorsions. Une façon d'atténuer ce phénomène serait donc de minimiser le déplacement de la courbure. Dans une paramétrisation globale, cela s'effectue en concentrant la courbure sur un ensemble réduis de points appelés *cônes*. Étant donné une distribution de cônes sur la surface, il est alors possible de les relier par des coutures, de sorte que, par construction, leur courbure se retrouve sur les limites de la carte. La surface peut ensuite être dépliée dans le plan sans déplacement supplémentaire de la courbure, et donc sans distorsion supplémentaire. La figure 1.1 illustre ce processus sur le modèle du *Lapin de Stanford*, tandis que la figure 1.4 montre une carte de la Terre calculée avec 8 cônes de courbure $\pi/2$. Dans ce contexte, le problème de la recherche d'une paramétrisation de bonne qualité a été transformé en recherche d'un ensemble approprié de cônes.

Paramétrisation de Maillages Surfaciques

Le calcul de paramétrisations de bonne qualité est devenu un problème central dans le domaine du traitement numérique de la géométrie, où la théorie et les résultats mathématiques concernant les surfaces lisses sont appliqués aux représentations numériques des données géométriques. Dans ce contexte, les maillages sont peut être les représentations de surfaces les plus répandues et populaires. En termes simples, un maillage est polyèdre, c'est à dire une surface faites de sommets, arêtes et faces, où chaque face est un polygone simple (Figure 1.6). Cette représentation offre un bon compromis entre la consommation de mémoire et la représentation précise d'une forme. Les maillages sont généralement représentés de manière extrinsèque, avec les coordonnées 3D de leurs sommets et quelques informations de connectivité pour les polygones. Pour un maillage surfacique, la recherche de la représentation 2D sous-jacente sous la forme d'une paramétrisation est utilisée comme étape de précalcul dans diverses applications [Botsch et al. 2010].

Plaquage de textures En informatique graphique, les maillages sont créés et utilisés par les artistes pour représenter les objets dans une scène virtuelle. Les couleurs et les textures sont définies sur ces maillages en en calculant une carte puis en y superposant une image plane. Ce processus, appelé "plaquage de texture" ou "cartographie *uv*" (Figure 1.5a), est au coeur de l'imagerie générée par ordinateur et est encore aujourd'hui considéré comme l'état de l'art. Dans ce contexte, la distorsion de la paramétrisation est directement visible sur la texture, et sa minimisation est donc cruciale.

Cartes de normales, ajout de détails Tout comme le plaquage de texture, la direction normale définie sur les faces d'un maillage peut être lissée et encodée sous forme d'image RVB superposée à une paramétrisation (avec 256 valeurs possibles pour les directions x, y et z). L'édition de cette texture normale modifiera le comportement des moteurs de rendu simulant la lumière sans qu'il soit nécessaire de modifier la géométrie (Figure 1.5b). Il est alors possible d'ajouter de la profondeur ou des détails à un modèle avec un minimum de calculs supplémentaires, ou d'appliquer la même carte de normales à un maillage décimé pour réduire le temps de rendu avec peu ou pas de dégradation visuelle. Ceci est particulièrement utile dans des contextes tels que les jeux vidéo où le budget temps par image est limité.



(a) Plaquage de texture sur le modèle de *Bob*, créé par Keenan Crane.

(b) Carte de normales du modèle *Nicolo da uzzano*.

Figure 1.5: Plaquage de texture (gauche) et carte de normales (droite) utilisées dans le logiciel de modélisation *Blender*.

Correspondances entre surfaces Dans les applications nécessitant de calculer une transformation entre deux surfaces, une paramétrisation peut être utilisée comme représentation commune intermédiaire [Schmidt et al. 2019]. Ce processus, connu sous le nom de paramétrisation croisée (*cross-parametrization*), permet des applications telles que le morphing [Kraevoy and Sheffer 2004], le transfert d'informations [Schmidt et al. 2020] ou même la classification des surfaces [Morreale et al. 2021], tout en restant assez peu sensible à la connectivité sous-jacente des maillages considérées.

Remaillage Enfin, contrairement au contexte théorique des surfaces lisses en mathématiques, l'utilisation de maillages dans la pratique introduit une structure combinatoire de sommets, d'arêtes et de faces. Deux maillages ayant une connectivité différente peuvent en effet représenter la même surface sous-jacente, mais la forme, la qualité et la nature de leurs faces polygonales jouent un rôle important dans la convergence, la stabilité et la précision des algorithmes de traitement, par exemple dans la simulation numérique [Schneider et al. 2022]. Dès lors, des

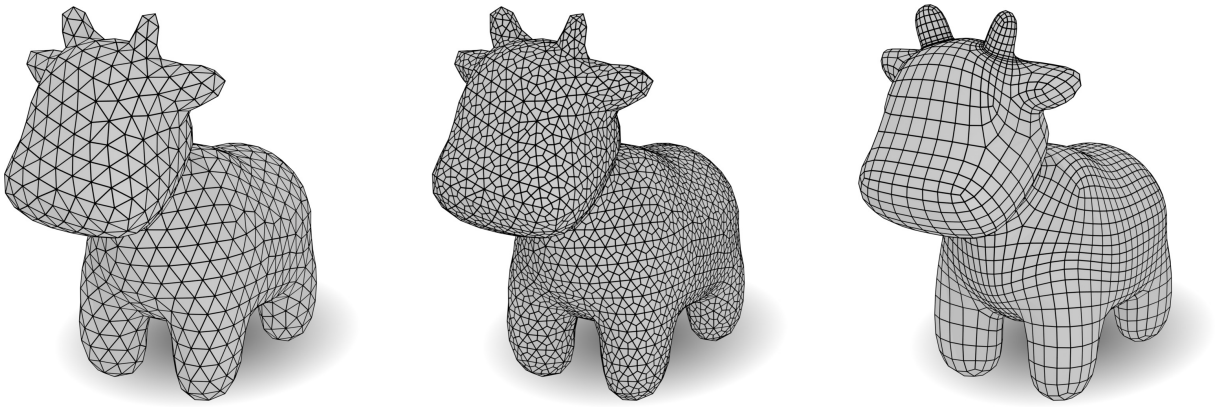


Figure 1.6: Trois maillages différentes du modèle *Spot* [Crane et al. 2013]. Gauche : maillage triangulé obtenu par la méthode de remaillage de Lévy and Liu [2010]. Milieu : Quadmesh irrégulier obtenu par subdivision du précédent. Droite : quadmesh original.

algorithmes de remaillage, améliorant la qualité de la représentation sous-jacente d'une surface, ont trouvé leur pertinence.

Alors que de nombreuses approches ont été explorées au fil des ans [Alliez et al. 2008; Yan et al. 2009] pour divers types maillages cibles, les méthodes basées sur la paramétrisation (comme [Remacle et al. 2009]) utilisent le fait que la paramétrisation d'un maillage est moins dépendante de sa combinatoire. Étant donné une carte de bonne qualité, une nouvelle structure sommets-arêtes-faces peut donc être dessinée directement dans l'espace paramétrique avant d'être ramenée à la surface d'origine [Botsch et al. 2010].

Ce type de technique peut être utilisé pour obtenir un maillage triangulé à partir d'un autre, mais aussi pour modifier la nature des faces à l'intérieur d'un maillage. Par exemple, lorsque la paramétrisation calculée satisfait certaines propriétés spécifiques, il est possible de remailler une surface triangulée avec des quadrilatères, créant ainsi un maillage quadrangulé (*quadmesh*).

Le problème exact du calcul d'un quadmesh à l'aide d'un algorithme efficace et polyvalent basé sur la paramétrisation est l'objectif de ce travail.

Des triangles aux quadrilatères

Les maillages triangulés et quadrangulés sont sans aucun doute les deux formes les plus répandues de maillages polygonaux. Comme tout polygone plan peut être décomposé en un ensemble de triangles, les premiers peuvent être considérés comme la représentation préférée des données géométriques. Mais dans certains contextes, les quadrangulations sont préférées en raison de leurs propriétés très particulières.

Premièrement, les artistes en informatique graphiques utilisent souvent des maillages quadrangulés en modélisation, car ces derniers peuvent être subdivisés plus naturellement autour de boucles de bord spécifiques ou à partir d'un ensemble grossier de points de contrôle. Par exemple, le quadmesh de droite de la figure 1.6 a été obtenu à partir d'un tel ensemble de contrôle fait à la main par le biais de deux itérations du schéma de subdivision de surface Catmull-Clark [Catmull and Clark 1998]. En outre, seuls les éléments quadrilatéraux peuvent être alignés partout avec la courbure d'une surface, ce qui aboutit à sa décomposition la plus naturelle. Il s'agit également de la décomposition avec le nombre minimal d'éléments nécessaires pour un niveau de détail donné [Alliez et al. 2003]. Lors de l'animation et de la déformation du maillage, le fait d'avoir

des arêtes et des quads orientés de cette manière minimise la déformation de chaque élément, réduisant ainsi le risque d'auto-intersections ou d'artefacts visuels.

Deuxièmement, dans le domaine de la simulation numérique, les maillages quadrilatéraux structurés représentent un compromis entre les grilles de voxels et les maillages triangulés, partageant la régularité des premiers tout en étant capables de se conformer aux limites du domaine considéré. Les schémas de subdivision de leur élément permettent de définir des quadrilatères à forte anisotropie lorsque la précision n'est requise que dans une seule direction, par exemple dans la simulation des couches limites en dynamique des fluides numérique [Garimella and Shephard 2000; Marcum et al. 2017]. Enfin, dans un contexte de simulation de déformations, l'erreur d'approximation est limitée par une fonction qui augmente avec l'angle minimal des éléments [Arnold et al. 2002; Ciarlet and Raviart 1972]. Étant donné que les maillages quadrangulés peuvent être réalisés avec des angles presque droits partout et étirés dans une direction avec des changements minimes de ces angles, ils permettent d'obtenir des résultats plus robustes qu'un maillage triangulaire, susceptible de présenter des triangles dégénérés.

Concevoir un algorithme générant automatiquement un maillage quadrangulé d'une surface donné est ainsi un défi majeur pour plusieurs domaines d'application. Cependant, la difficulté ne réside pas dans la génération de n'importe quel maillage – on peut toujours extraire un quadmesh d'une triangulation en décomposant chaque triangle en trois quadrilatères relié à un point central (Figure 1.6, milieu) – mais bien le calcul d'un quadmesh présentant les propriétés d'alignement ou d'anisotropie mentionnées précédemment. En particulier, il est généralement préférable de calculer un quadmesh présentant une certaine régularité, avec la majorité de ses sommets adjacents à 4 quads et seulement une poignée d'entre eux adjacents à 3 ou 5 quads. Ce problème de la génération de maillages quadrangulés de bonne qualité a suscité d'abondantes recherches au cours des 20 dernières années [Bommes et al. 2013b].

Une approche prometteuse au problème du remaillage quad s'appuie sur une paramétrisation du domaine initial. En se basant sur l'observation qu'un tel maillage consiste en une déformation de la grille euclidienne presque partout sauf en ses points irréguliers, l'idée de l'approche basée paramétrisation est donc de superposer la grille à une représentation plane du domaine pour ensuite la reprojeter sur la surface et ainsi en obtenir une décomposition en quadrilatères (Figure 1.1). Toutefois, pour que cette méthode soit efficace, la paramétrisation calculée doit satisfaire un ensemble de constraints très spécifiques, notamment pour que les lignes de quads restent continues au travers des coutures. On parle alors de paramétrisation *sans coutures* (*seamless*) ou de paramétrisation qui préserve la grille.

Le calcul d'une paramétrisation sans coutures ajoute un autre niveau de complexité au problème de calcul de cartes de surface. Comme une telle paramétrisation est définie par un ensemble de cônes de singularité dont la courbure est verrouillée pour être un multiple de $\pi/2$ ainsi que par un ajustement précis des arêtes correspondantes à travers les coutures, le problème d'optimisation à résoudre implique des variables entières, le rendant notoirement difficile à résoudre en pratique. Pour obtenir des résultats en pratique, les méthodes de la littérature le décomposent généralement en un pipeline de plusieurs problèmes d'optimisation plus faciles, chacun satisfaisant une partie des contraintes nécessaires [Bommes et al. 2009].

La première étape d'un tel pipeline consiste à déterminer la position des cônes de singularité, qui correspondront aux sommets irréguliers dans le maillage final. Ensuite, une paramétrisation conforme à cette distribution de cônes est calculée. Ensuite, les coordonnées des sommets dans le plan sont ajustées pour assurer la continuité des lignes du quadrillage. Enfin, un quadmesh valide est extrait [Ebke et al. 2013].

Bien que cette décomposition du problème ait atteint l'état de l'art dans les techniques de génération de quadmesh, la division du calcul en plusieurs problèmes d'optimisation n'a pas été

sans inconvénients. Comme chaque étape détermine des degrés de liberté distincts, les décisions prises à un moment donné ne peuvent pas être inversées, ce qui peut conduire à des résultats sous-optimaux, voire à des cas d'échec. Par exemple, il se peut qu'il n'existe pas de paramétrisation sans couture qui soit à la fois injective et conforme à la distribution calculée des cônes. De plus, bien que des recherches approfondies aient été menées pour améliorer chaque étape, les études sur leurs interactions et leurs effets sur le quadmesh final sont encore rares.

Objectifs de cette thèse

L'objectif de cette thèse est d'améliorer le pipeline de paramétrisation sans couture pour la production automatique de maillages quadrangulés, dans l'optique de résoudre les problèmes évoqués un peu plus haut. Plus spécifiquement, notre travail se concentre sur les liens entre les cônes, la paramétrisation et les propriétés du quadmesh résultant.

Notre approche s'appuie sur la conception d'algorithmes permettant de mieux contrôler les degrés de liberté d'une paramétrisation sans couture, en particulier sa distribution de cônes et sa distorsion. En pratique, nous regroupons plusieurs étapes du pipeline en un seul problème d'optimisation, tirant ainsi pleinement parti de compromis inaccessibles dans le pipeline actuel.

Un effort particulier a été fourni pour formuler ces problèmes d'optimisation en termes de variables continues, évitant ainsi les formulation en nombre entiers des travaux précédents qui peuvent s'avérer difficile à résoudre.

Importantly, we formulate all of our optimization problems in terms of continuous variables, thus avoiding the difficult mixed-integer formulations of some previous works.

Enfin, une motivation centrale à notre travail réside dans sa potentielle généralisation au cas volumique du maillage hexaédrique. La génération de maillages hexaédriques robustes basée sur des approches de paramétrisation est en effet encore hors de portée des techniques actuelles, de nombreux algorithmes aptes pour les surfaces ne généralisant en effet pas à la dimension supérieure, pour des raisons à la fois théoriques et pratiques. Bien que nous ne fassions pas de contributions explicites au domaine du maillage hexagonal dans ce manuscrit, l'application de nos méthodes au cas 3D reste un travail futur prometteur.

Organisation du manuscrit

La suite de ce manuscrit est divisée en six chapitres.

Le chapitre 2 introduit le contexte mathématique nécessaire à la formulation de notre problème et donne un aperçu de l'état de l'art en matière de paramétrisation de surface.

Dans le chapitre 3, nous exposons une première méthode de calcul d'une distribution de cônes aux défauts d'angle multiples de $\pi/2$ et d'une paramétrisation sans couture associée. Pour éviter toute variable entière dans la formulation, nous utilisons une astuce combinatoire et définissons un autre domaine sur lequel l'optimisation devient continue.

Dans le chapitre 4, nous nous inspirons de méthodes de géométrie différentielle, plus particulièrement de la méthode des repères mobiles d'Elie Cartan, pour concevoir un algorithme qui optimise simultanément une distribution de cônes et une paramétrisation sans coutures, ce qui permet un contrôle étendu sur le résultat final. Le chapitre 5 présente une variante du même algorithme avec moins de garanties théoriques mais de meilleures performances en pratique.

Dans le chapitre 6, nous exposons nos travaux en cours sur la généralisation de la méthode précédente des repères mobiles au cas des paramétrisations préservant la grille, concevant ainsi un problème d'optimisation qui résout chaque étape du pipeline en une seule fois.

Enfin, notre chapitre conclusif (Chapitre 7) résume nos contributions et discute de leurs extensions, à la fois pour le maillage quadrangulé mais aussi leur généralisation au maillage hexaédrique.

Publications associées

Publications scientifiques

Les travaux présentés dans ce manuscrit, en particulier le Chapitre 4, ont fait l'objet d'une publication dans le journal *Transaction on Graphics* in 2023:

- *The Method of Moving Frames for Surface Global Parametrization*, **Guillaume Coiffier** and Etienne Corman, Transaction on Graphics, 2023

Bien que n'ayant aucun lien avec le sujet des cartes sans coutures pour le maillage quad, nous avons aussi contribué durant cette thèse à d'autres projets de recherche en apprentissage profond et traitement de la géométrie :

- *Parametric surface fitting on airborne lidar point clouds for building reconstruction*, **Guillaume Coiffier**, Justine Basselin, Nicolas Ray and Dmitry Sokolov, Symposium on Solid and Physical Modeling, 2021
- *Robust One-Class Classification with Signed Distance Function using 1-Lipschitz Neural Networks*, Louis Béthune, Paul Novello, Thibaut Boissin, **Guillaume Coiffier**, Mathieu Serrurier, Quentin Vincenot and Andres Troya-Galvis, International Conference on Machine Learning, 2023

Logiciels

La plupart de notre code de gestion de maillages, ainsi que les implémentations de méthodes de paramétrisations de la littérature utilisées pour les figures de ce manuscrit, ont été regroupée en une librairie Python appelée *mouette* : `https://github.com/GCoiffier/mouette`.

L'implémentation de notre méthode de paramétrisation basée sur les repères mobiles de Cartan des Chapitres 4 et 5 a également été implémentée en python en se basant sur *mouette* et peut être téléchargée au lien suivant : `https://github.com/GCoiffier/moving_frames_parametrization`

1

Introduction

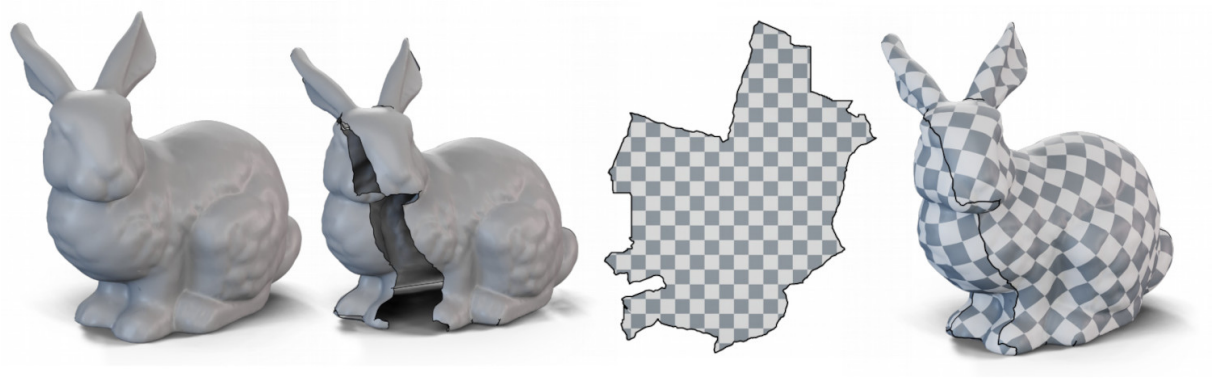


Figure 1.1: Global parametrization of the *Stanford Bunny*. A surface represented as a triangle mesh (left) is cut along a set of seams (middle left) and flattened to the plane with minimal distortion (middle right). A regular grid, represented here as a checker texture can be superimposed on this flat representation and lifted back onto the initial surface (right).

Maps, Seams and Curvature

Geometry is the study of shapes. Its mathematical formalism allows us to understand the objects surrounding us, from the cosmic-scale effects of general relativity, to the curviness of the Earth, to the most mundane objects like sewed clothes or candy wrappers, enabling many breakthroughs throughout science and technology.

How to mathematically describe a shape? Suppose for instance that we are given a smooth surface in three-dimensional space. One possibility to describe it mathematically is to exhibit the set of all points that belong to the surface. While it is technically a valid description, this says nothing about its structure. A surface is indeed fundamentally a two-dimensional object, meaning that the trajectory of a point traveling on it is restrained to two degrees of freedom: "forward/backward" and "left/right". From this local point of view, a surface behaves like the classical euclidean plane. This means that one should be able to precisely describe the position of a point by providing how much "left" and "forward" to travel in order to reach this point from an arbitrary origin. In other words, this is equivalent to assigning a consistent two-dimensional

system of coordinates to the surface, that is to say unfolding the surface and laying it flat. This process is called a surface *parametrization*.

As the parametrization of a surface is simply a representation of said surface into the plane, it is easy to find connections with day-to-day tasks. The first and perhaps most obvious of these tasks is the fabrication of geographical maps, that is to say an easy-to-grasp representation of the surface of the Earth. As the Earth is generally identified with a perfect sphere, obtaining a parametrization can be and has been done by hand in many different ways, often focusing on geopolitical constraints rather than mathematical ones.

Another task closely related to surface parametrization is the fabrication of clothes, where initially flat pieces of fabric need to be seamed to form a non-flat object in the end. But again, as clothes generally present a common overall shape, this decomposition into fabric patterns (front, back, sleeves, and so on) have been done without the help of computers for centuries, although automation in garment manufacturing and design is an active area of research [Nayak and Padhye 2017].

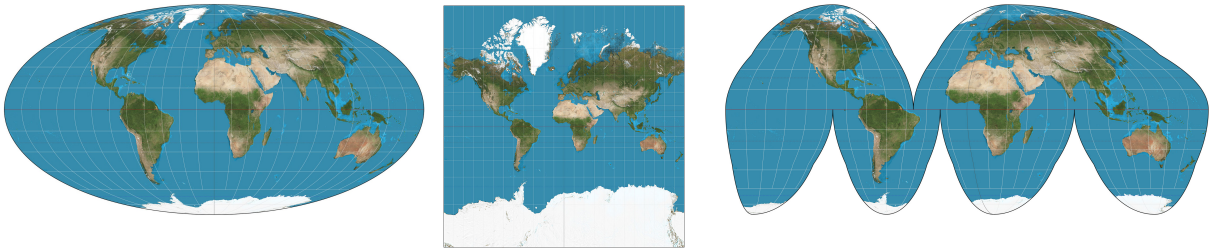
It is only more recently, in the last 50 or so years, that the apparition of 3D computer graphics stemmed the need for automatic algorithms for surface parametrization. In video games and animated films applications, computing and storing a flat representation of a given object is indeed the canonical technique to give it colors and details in the rendered images. Due to the variety and number of objects to be processed in a single scene, automation of the parametrization step has become an essential time and quality gain.

In all of these applications, it is not only required to compute a map of an object, but to also meet some constraints. To be useful, maps should indeed exhibit some natural properties. The first one is that the surface must be represented as accurately as possible. In other words, a given shape drawn on the surface should not be distorted when flattened, both in terms of size (scaling) and deformation (shearing, stretching). In the case of geographical maps, one would indeed expect landmasses and oceans to have recognizable shapes and their relative areas to stay constant. Unfortunately, this turns out to be impossible: a so-called *isometric* mapping of the Earth cannot mathematically exist. One has to resort to a tradeoff between perfectly preserving relative areas (like the Mollweide projection of Figure 1.2a) at the expense of stretching, or perfectly preserving shapes (like the Mercator projection, Figure 1.2b) at the expense of sometimes dramatic changes in relative areas. As of 2023, the Wikipedia page listing types of projection for Earth maps[6] contains more than 80 entries, illustrating the need for different maps depending on the targeted application.

Another important property of surface parametrizations is injectivity. Simply put, each point of the map should correspond to as most one point on the surface. Indeed, a map that puts Paris and Tokyo at the same point is likely to be impossible to read. This has one important consequence: depending on the topology of the considered surface, it may be necessary to perform cuts to be able to correctly unfold the surface. In the case of a sphere like Earth, it is usual to perform a cut from the north to the south pole going through the ocean. Likewise, a garment like a T-shirt can only be crafted from a piece of material where the sleeves have been cut as rectangles before being seamed in a cylindrical shape. From this last example comes the common denomination of cuts in a surface parametrization: seams.

Seams on a parametrization are not only necessary due to topological reasons but they can also be added to minimize distortion. This idea comes from a physical intuition: when we consider the flattening of a real world object like the peel of a tangerine (Figure 1.3), the stiffness of the material prevents arbitrary large deformations. Instead, the object is more likely to tear when

[6] `https://en.wikipedia.org/wiki/List_of_map_projections`



(a) Mollweide (b) Mercator (c) Goode homolosine

Figure 1.2: Different projections of the Earth's surface to the plane. (a) The area-preserving Mollweide projection perfectly preserves the relative sizes of landmasses. (b) The *conformal* Mercator projection preserves angles and shapes at the expense of dramatic scaling at the poles. While it is mathematically impossible for a mapping to preserve both, this optimum can be approached by introducing cuts (c). Images taken from *Wikipedia*[6].

flattened, creating a seam, and the resulting parametrization is likely to be less distorted overall (as with the Goode projection of Figure 1.2c). This opens another tradeoff between the number of discontinuities and the distortion of the mapping.

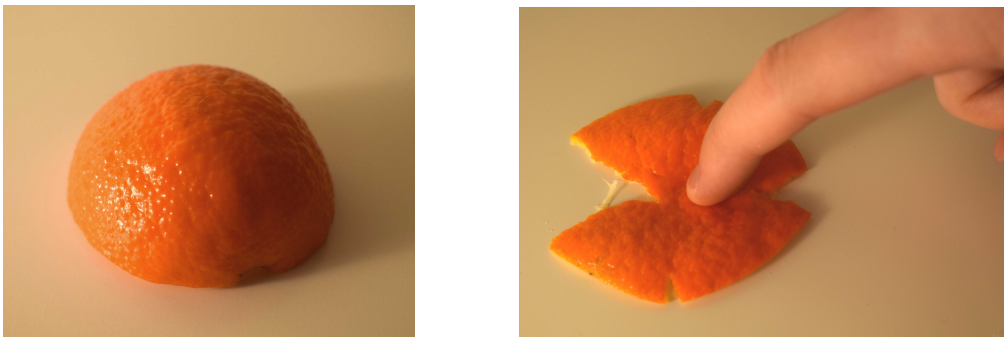


Figure 1.3: In the real world, trying to flatten an object usually results in tears, like this peel of tangerine. Tearing happens naturally as a way to alleviate distortion.

Now, given an arbitrary smooth surface as an input, computing a good quality parametrization requires to answer a number of questions: Should local areas or shapes be preserved ? It is better to have a connected rectangular map with high distortion, or a lower distortion maps with many seams ? Given a surface, what is its best possible parametrization under some distortion or boundary criterion ? In addition, some applications, like the one we are interested in for this work, require additional constraints onto the final mapping, such as vertical or horizontal alignment of the map's boundary or controlled distortion across seams, making the problem of finding an optimal parametrization even more challenging.

The concept of *global* parametrization is a promising step towards answering these questions. It is based on the key observation that distortion is caused by a change of curvature. Curvature, more precisely Gaussian curvature, is a number that quantifies how a surface locally differs from a flat one. Since it can be arbitrarily distributed on a surface but is only present on the boundaries of its flat representation, any parametrization mapping has to displace it in some ways and unavoidably create distortion in the process. One way to alleviate this phenomenon is to minimize the displacement of curvature. In a global parametrization, this can be done by concentrating curvature onto a finite, often small set of points called *cones*. Given a cone

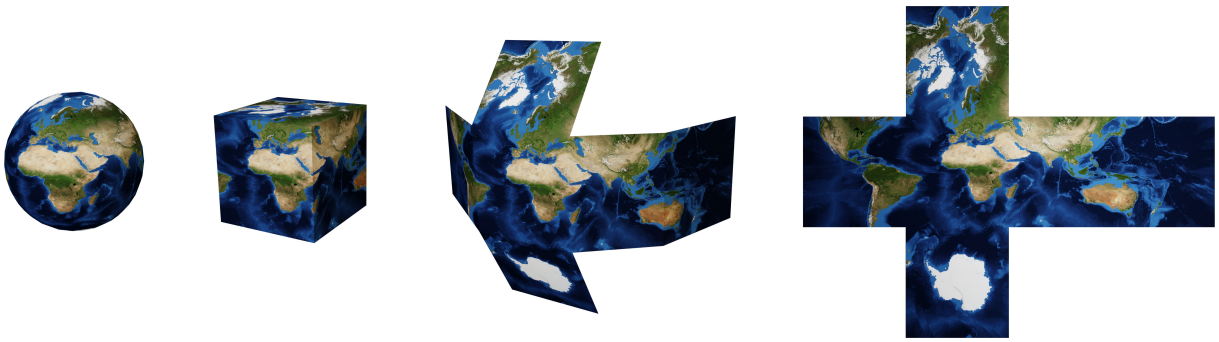


Figure 1.4: A global parametrization maps the curvature distribution of a smooth surface to a finite set of cones. Here, 8 cones each concentrate $\pi/2$ as curvature, thus leading to a mapping of the Earth to a cube. The parametrization is then obtained by cutting the cube between cones and unfold the pattern without further distortion.

distribution on the surface, we can then link them by seams so that curvature now already lays only on the boundary of the shape by construction. The surface can therefore be mapped in the plane with no additional curvature displacement, and thus no additional distortion. Figure 1.1 illustrates this process on the *Stanford Bunny* model, while Figure 1.4 shows an Earth map computed with 8 cones of curvature $\pi/2$. In this context, the problem of finding a good quality parametrization has been transformed to finding a suitable set of cones.

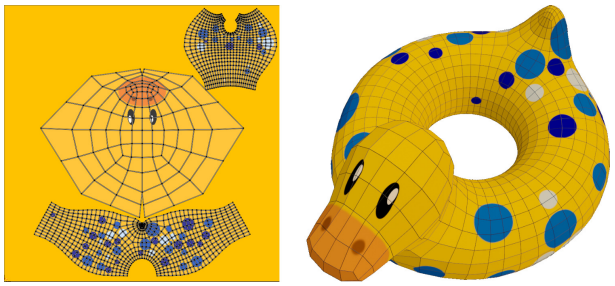
Parametrization of Surface Meshes

The computation of good quality parametrizations has become a central problem in the field of *Digital Geometry Processing*, where theory and results on smooth manifolds are applied to digital representations of geometrical data. The most widespread and perhaps most successful of such representation is the mesh. Simply put, a mesh is the partition of a 2D domain using simple polygons (Figure 1.6). It provides a good tradeoff between memory consumption and accurate representation of the shape. Meshes are generally represented extrinsically, with the 3D coordinates of their vertices and some connectivity information for polygons. For a surface mesh, finding the underlying 2D representation in form of a parametrization is used as a precomputing step in various applications [Botsch et al. 2010].

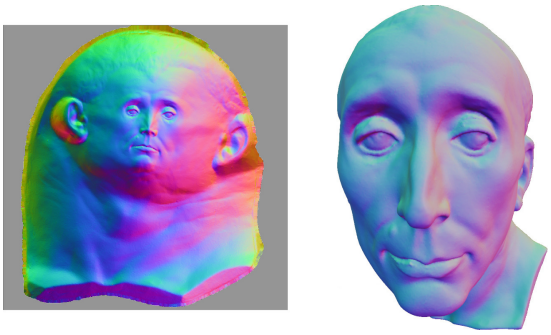
Texture Mapping In computer graphics, meshes are created and used by artists to represent objects in a virtual scene of an animated film or a video game. Colors and texture are defined on these meshes by mapping the mesh to the plane and overlay a flat image texture onto the parametrization. This process called texture mapping or *uv*-mapping (Figure 1.5a) is at the heart of computer-generated imagery and still considered state of the art to this day. In this context, the distortion of the parametrization is directly visible onto the mapped texture, hence its minimization is crucial.

Normal mapping and detailing Just like texture mapping, the normal direction defined over the faces of a mesh can be smoothed and encoded as a RGB image overlaid on a parametrization (with 256 possible values for x,y and z directions). Editing this normal texture will modify the behavior of light-simulating render engines without needing any change in the geometry (Figure 1.5b). It is then possible to add depth or details to a model with minimal additional

computations, or apply the normal map to a decimated mesh to cut rendering time with little to no visual degradation. This is especially useful in contexts like video games where time budget per frame is limited.



(a) Texture mapping on Keenan Crane's *Bob* model.



(b) Normal mapping of the *Nicolo da uzzano* head model.

Figure 1.5: Texture mapping (left) and normal mapping (right) as in the *Blender* software.

Inter-surface maps In applications where correspondences or explicit mapping between two surfaces are involved, a parametrization can be used as an intermediate common representation [Schmidt et al. 2019]. This process, known as cross-parametrization enables applications like morphing [Kraevoy and Sheffer 2004], information transfer [Schmidt et al. 2020] or even surface classification [Morreale et al. 2021], while staying agnostic to the underlying connectivity of the considered meshes.

Remeshing Finally, unlike the theoretical context of smooth surfaces in mathematics, the use of meshes in practice introduces a combinatorial structure of vertices, edges and faces. Two meshes with a different connectivity can indeed represent the same underlying surface, yet the shape, quality and nature of their polygonal faces play a big role in the convergence, stability and precision of processing algorithms, for instance in numerical simulation [Schneider et al. 2022].

While many approaches have been explored throughout the years [Alliez et al. 2008; Yan et al. 2009] for various target meshes, methods based on parametrization (like [Remacle et al. 2009]) use the fact that the parametrization of a mesh is less dependent on its combinatorics. Given a good quality mapping, a new vertex-edge-face structure can therefore be drawn directly into parameter space before being lifted back to the original surface [Botsch et al. 2010].

This type of technique can be used to obtain a triangular mesh from another, but also to change the nature of faces inside a mesh. For instance, when the computed parametrization satisfies some specific properties, it is possible to remesh a triangulated surface with quadrilaterals, creating a *quadmesh*.

The exact problem of computing a quadmesh using efficient and versatile parametrization-based algorithm is the goal of this work.

From Triangles to Quadrilaterals

Triangular and quadrilateral meshes are without a doubt the two most widespread forms of polygonal meshes. As any planar polygon can be decomposed into a set of triangles, the first

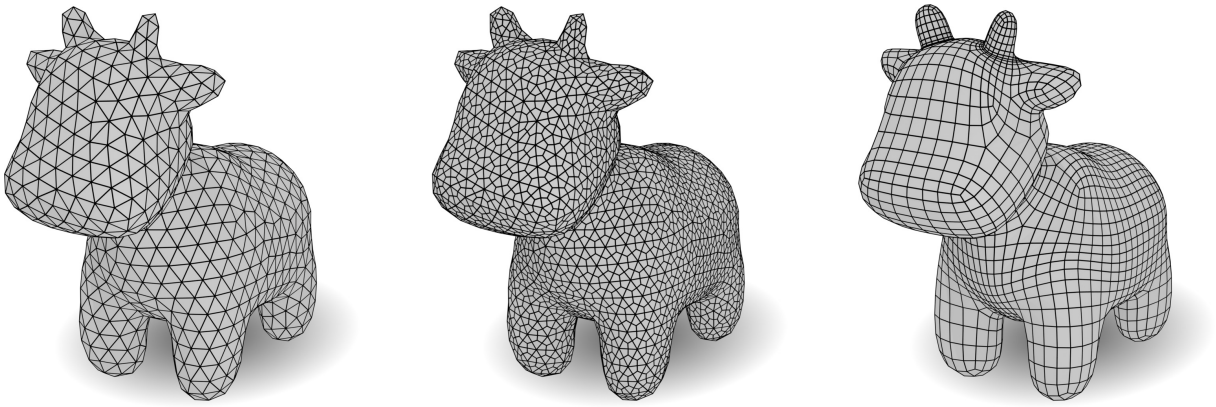


Figure 1.6: Three different meshes of the *Spot* model [Crane et al. 2013]. Left: triangular mesh obtained via the method of Lévy and Liu [2010]. Middle: irregular quadmesh obtained from the latter. Right: original structured quadmesh.

ones could be considered as the preferred representation of geometrical data. Yet, quadmeshes are often preferred to their triangular counterpart in various contexts, due to their very particular properties.

Firstly, artists in computer graphics often rely on quadmeshes for modeling because they can be more naturally subdivided around specific edge loops or from a coarse set of control points. For instance, the right quadmesh of Figure 1.6 was obtained from such a handmade control set made via two iterations of the Catmull-Clark surface subdivision scheme [Catmull and Clark 1998]. Moreover, only quadrilateral elements can be aligned everywhere with the curvature of a surface, which results in its most natural decomposition. This is also the decomposition with the minimal number of elements needed for a given level of detail [Alliez et al. 2003]. When dealing with animation and deformation of the mesh, having edges and quads oriented that way minimizes the amount of deformation on specific elements thus reducing the risk of auto-intersection or visual artifacts.

Secondly, in the domain of numerical simulation, structured quadmeshes represent a tradeoff between voxel grids and triangle meshes, as they share the regularity of the former while being able to conform to the boundary of the domain. Subdivision schemes over their element allow defining quadrilaterals with high anisotropy where precision is required in only one direction, for instance in the simulation of boundary layers in computational fluid dynamics [Garimella and Shephard 2000; Marcum et al. 2017]. Finally, in a context of deformation simulation, it is known that the approximation error is bounded by a function that increases with the minimal angle of the elements [Arnold et al. 2002; Ciarlet and Raviart 1972]. Since quadmeshes can be made with almost right angles everywhere and stretched in a direction with minimal changes to these angles, a quadmesh will allow more robust results than a triangle mesh which is likely to present degenerated triangles.

Designing algorithms to automatically generate a quadmesh of a given surface is therefore a major challenge for a variety of applicative domains. However, the difficulty does not lie in the generation of any quadmesh – one can always be computed from a triangle mesh by splitting each triangle into three quadrilaterals from a central point (middle of Figure 1.6) – but the computation of one with the aforementioned properties of alignment or anisotropy. In particular, it is usually desirable to compute a quadmesh exhibiting some regularity, with most of its vertices being adjacent to 4 quads and only a handful of them being adjacent to 3 or 5.

This specific problem has sparked abundant research for the last 20 years [Bommes et al. 2013b].

A promising approach to the problem of quadmeshing utilizes a parametrization of the initial domain. Based on the observation that a semi-regular quadmesh consists in a deformation of the Euclidean grid everywhere except at its irregular points, the idea behind the parametrization approach is to therefore overlay the grid over a flat representation of the domain and lift it back onto the surface to obtain a quad decomposition (Figure 1.1). However, for this method to be successful, the parametrization computed should satisfy a very specific set of constraints, in order for lines of quads to stay continuous across the (necessary) seams. We speak of *seamless* or *grid-preserving* parametrizations.

Computing a seamless parametrization adds another level of complexity over the problem of mapping a surface into the plane. As such a parametrization is defined by a set of singularity cones with curvature locked to be a multiple of $\pi/2$ as well as precise adjustment of matching edges across seams, the optimization problem involved require integer variables which make them notoriously hard to solve in practice. In order for the optimization to be tractable, state-of-the-art methods usually divide the computation in a pipeline of several easier optimization problems, each satisfying the needed constraints one at a time [Bommes et al. 2009].

The first step in such a pipeline is to determine the position of singularity cones, which will correspond to irregular vertices in the final quadmesh. Then, a parametrization conforming to this topology is computed. After that, coordinates of vertices in the plane are adjusted to ensure the continuity of quad lines. Finally, a valid quadmesh is extracted [Ebke et al. 2013].

While this pipelined approach settled at the state of the art in quadmeshing techniques, splitting the computation into several optimization problems did not come without drawbacks. As each step determines distinct degrees of freedom, decisions made at one point cannot always be reversed, leading to suboptimal results or even failure cases. For example, a seamless parametrization that is both injective and conforming to the computed distribution of cones may not exist. Furthermore, extensive research has been conducted for the improvement of each steps, but the study of their interaction and their effects onto the final quadmesh – arguably a more difficult problem to tackle – is still sparse.

Objectives of this work

The goal of this thesis is to improve the seamless parametrization pipeline for the production of quadmeshes, in order to tackle the aforementioned issues. More specifically, our work focuses on the links between the cones, the parametrization and the resulting quadmesh properties. We attempt at designing algorithms with more control over the degrees of freedom of a seamless parametrization, in particular its cone distribution and its distortion. We design optimization algorithms to solve for several of these variables in a single step, thus taking full advantage of tradeoffs inaccessible in the current pipeline. Importantly, we formulate all of our optimization problems in terms of continuous variables, thus avoiding the difficult mixed-integer formulations of some previous works.

Finally, a key motivation of our work lies in its possible generalization to the volume case of hexmeshing. Robust hexahedral mesh generation based on parametrization approaches is still out of reach as many algorithms for surfaces do not generalize to the upper dimension both due to theoretical and practical reasons. While we do not make explicit contributions to the domain of hexmeshing in this manuscript, the application of our methods to the 3D case remains a promising future work.

Organization of this manuscript

The rest of this manuscript is divided into 6 chapters.

Chapter 2 will introduce the mathematical background needed to formulate our problem and will give an overview of the state of the art in surface parametrization.

In Chapter 3, we will focus on a first method to compute a cone distribution where defects are bound to be multiples of $\pi/2$ and recover an associated seamless parametrization. To circumvent the problem of integer variables, we use a combinatorial trick and define another domain over which the optimization becomes continuous.

In Chapter 4, we take inspiration from the differential geometry literature, more specifically Elie Cartan's method of moving frames, to design an algorithm that simultaneously optimizes for a cone distribution and a parametrization, enabling extensive control over the final result. Chapter 5 will present an alternative optimization of the same problem with less theoretical guarantees but better performance in practice.

In Chapter 6, we expose our ongoing work about generalizing the previous method of moving frames to the case of grid-preserving parametrizations, thus designing an optimization problem that solves every step of the pipeline in one go.

Finally, our conclusive chapter (Chapter 7) will summarize our contributions and discuss their extensions, both for quadmeshing in the surface case and their generalization to hexmeshing.

Associated material

Publications

The work presented in this manuscript, in particular Chapter 4, has been the object of a publication in *Transaction on Graphics* in 2023:

- *The Method of Moving Frames for Surface Global Parametrization*, **Guillaume Coiffier** and Etienne Corman, Transaction on Graphics, 2023

While unrelated to the topic of seamless parametrization, we contributed to other research works in geometry processing and deep learning during this PhD:

- *Parametric surface fitting on airborne lidar point clouds for building reconstruction*, **Guillaume Coiffier**, Justine Basselin, Nicolas Ray and Dmitry Sokolov, Symposium on Solid and Physical Modeling, 2021
- *Robust One-Class Classification with Signed Distance Function using 1-Lipschitz Neural Networks*, Louis Béthune, Paul Novello, Thibaut Boissin, **Guillaume Coiffier**, Mathieu Serrurier, Quentin Vincenot and Andres Troya-Galvis, International Conference on Machine Learning, 2023

Software

Most of our mesh processing code, as well as implementations of state of the art parametrization methods used for figures in this manuscript have been gathered into a single Python library called *mouette*: `https://github.com/GCoiffier/mouette`.

The implementation of the moving frame algorithms of Chapters 4 and 5, implemented in python using *mouette* can be found here: `https://github.com/GCoiffier/moving_frames_parametrization`

2

Background and State of the Art in Surface Parametrization

The field of geometry processing lies at the intersection of two vast and complementary areas of research. On one side, geometrical objects have been studied – and are still studied – as abstractions through the lens of mathematics and especially differential geometry. On the other side, the question of how to represent these objects inside a computer and how to design efficient algorithms to deal with them is motivated by modern applications in computer graphics and numerical simulation.

In this first chapter, we expose the required notions of both differential geometry (Section 2.1) and computer science (Section 2.2) needed to understand surface parametrization. We will then give an overview of the domain, following works from two of its most widespread applications : texture mapping and quadmeshing. The first one (Section 2.3) will allow us to adopt an historical point of view, from the early works of Tutte [1963] to state of the art methods and will introduce the main concepts. The second (Section 2.4) is the application we will be interested in for the rest of the manuscript.

Contents

2.1 Elements of Differential Geometry 20
2.1.1 Differentiable Maps 20
2.1.2 Manifolds 21
2.1.3 Vector Fields 21
2.1.4 Differential Forms 23
2.1.5 Parallel Transport 24
2.1.6 Holonomy and Curvature 25
2.1.7 Genus, Non-contractible Cycles and the Gauss-Bonnet Theorem 27
2.2 Surface Meshes 29
2.2.1 Definition of a Mesh 30
2.2.2 Discrete Differential Geometry 31
2.2.3 Parametrization of a Mesh 35
2.3 Surface Parametrization for Texture Mapping 37
2.3.1 Tutte's Embedding 37
2.3.2 Free-boundary Parametrization, Conformal Maps 39
2.3.3 Foldover-free Maps 41

2.3.4 Distortion Minimization 42
2.3.5 Introducing Cuts 44
2.3.6 Beyond Disk Topology 45
2.3.7 Making Seams Disappear 47
2.4 Seamless and Grid-preserving Parametrization Applied to Quadmeshing 50
2.4.1 Overview of Quad Generation Methods 50
2.4.2 The Parametrization-based Approach to Quadmeshing 52
2.4.3 Computing a Cone Distribution using Smooth Frame Fields 57
2.4.4 Rotationally Seamless Parametrization 62
2.4.5 Grid-preserving Parametrization 65

2.1 Elements of Differential Geometry

Let us start by exposing and defining the different concepts of differential geometry used throughout the manuscript. In a nutshell, differential geometry is the geometrical study of smooth shapes, called manifolds, and objects that are defined on them, such as functions or vector fields. While the whole theory can be derived in a very general and abstract way for any topological space, such an exposition is out of scope for this manuscript. While we give here some definitions and properties for an arbitrary finite dimension n, we will only consider in this work manifolds of dimension 2 that can be immersed in $\mathbb{R}^3$, that is to say smooth *surfaces*. The ability to use an external space $\mathbb{R}^3$ will simplify some concepts and allows us to better stick to the intuition one can have about those objects.

Notions exposed in this section are based on the textbooks of Morita [2001], Aubin [1998],do Carmo [1992] and O'Neill [2006].

2.1.1 Differentiable Maps

At the heart of differential geometry is the concept of continuous and differentiable maps. Such a map allows us to generalize the geometry of vector spaces to their images, which can be arbitrary surfaces, volumes and more.

Let U, V be two open sets of $\mathbb{R}^n$. A map $\varphi : U \to V$ is said to be

- an *homeomorphism* if it is bijective and both φ and φ^{-1} are continuous;
- a $\mathcal{C}^k$*-diffeomorphism* is it is bijective and both φ and φ^{-1} are of class $\mathcal{C}^k$.

If $\varphi : \mathbb{R}^m \to \mathbb{R}^n$ is a differentiable function, the *Jacobian Matrix* $J(\varphi)$ of φ is the $n \times m$ matrix of its partial derivatives with respect to a coordinate system $(x_1, ..., x_m)$ of $\mathbb{R}^m$:

$$J(\varphi) = \begin{pmatrix} \frac{\partial \varphi_1}{\partial x_1} & \frac{\partial \varphi_1}{\partial x_2} & \dots & \frac{\partial \varphi_1}{\partial x_m} \\ \frac{\partial \varphi_2}{\partial x_1} & \frac{\partial \varphi_2}{\partial x_2} & \dots & \frac{\partial \varphi_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \varphi_n}{\partial x_1} & \frac{\partial \varphi_n}{\partial x_2} & \dots & \frac{\partial \varphi_n}{\partial x_m} \end{pmatrix}$$

where $\frac{\partial \varphi_i}{\partial x_j}$ is by definition the partial derivative of the i-th coordinate of φ with respect to the j-th coordinate of $\mathbb{R}^n$.

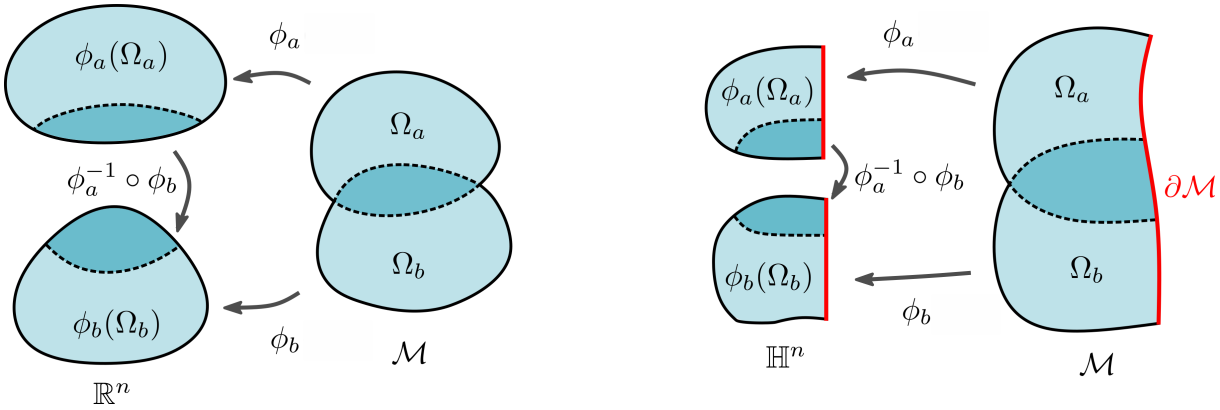


Figure 2.1: A parametrization consists of local coordinate systems ϕ defined over charts Ω such that the transition from one chart to the next is a $\mathcal{C}^k$ diffeomorphism. For manifolds with boundaries (right), we restrict our coordinate systems to be defined on a half-space $\mathbb{H}^n$ rather than $\mathbb{R}^n$.

2.1.2 Manifolds

A manifold $\mathcal{M}$ of dimension n is a set such that each point of $\mathcal{M}$ admits a neighborhood that is homeomorphic to $\mathbb{R}^n$. If each point of $\mathcal{M}$ admits a neighborhood that is $\mathcal{C}^k$-diffeomorphic to $\mathbb{R}^n$, we say that $\mathcal{M}$ is smooth of class $\mathcal{C}^k$.

A *local chart* of $\mathcal{M}$ is a pair (Ω, ϕ) where Ω is an open subset of $\mathcal{M}$ and $\phi : \Omega \to \mathbb{R}^n$ is a homeomorphism. The mapping ϕ is called the *parametrization* of local chart Ω. In 2D, we can think of it as the mapping that "unwraps" a part of the manifold and flattens it to the plane. This mapping is exactly what we aim at computing in practice in this work.

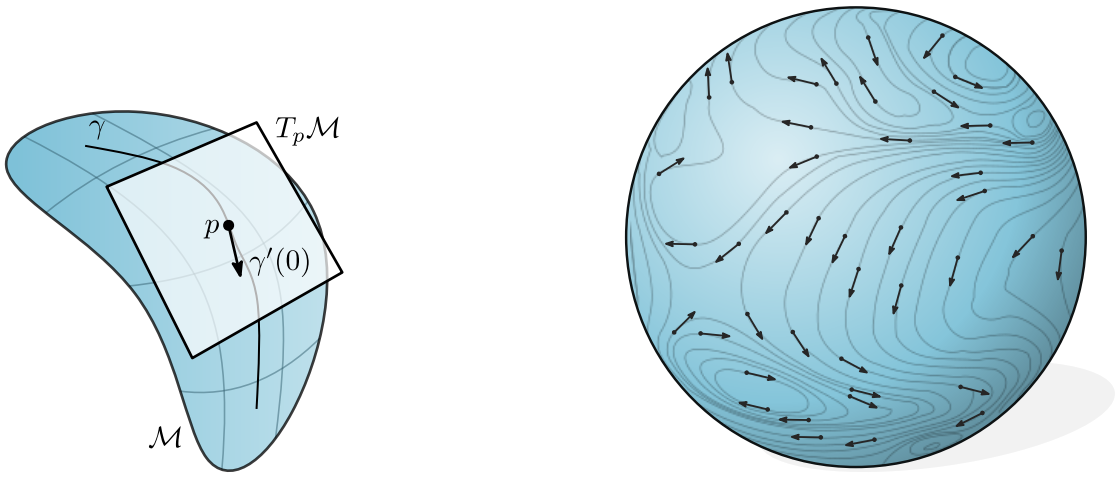
A collection $(\Omega_i, \phi_i)_{i \in I}$ of local charts such that $\mathcal{M} = \bigcup_i \Omega_i$ is called an *atlas*. As parametrizations ϕ_i over charts define local coordinate systems, an important and desirable property is that these coordinate systems are consistent, meaning that the transition from one to another can be done in a smooth manner. To this end, we say that an atlas is of class $\mathcal{C}^k$ if, for every pair of charts $(\Omega_a, \phi_a), (\Omega_b, \phi_b)$, the application $\phi_b \circ \phi_a^{-1} : \phi_a(\Omega_a \cap \Omega_b) \to \phi_b(\Omega_a \cap \Omega_b)$ is a $\mathcal{C}^k$-diffeomorphism. This property is illustrated on Figure 2.1 (left).

Boundary of a manifold For some manifolds, the previous definition has to be extended to take into account the presence of a boundary, since points on the boundary of $\mathcal{M}$ do not admit a neighborhood that can be parametrized to an open subset of $\mathbb{R}^n$. To take these points into account, it is sufficient to consider that the manifold can be locally diffeomorphic to either $\mathbb{R}^n$ (for interior points) or the half-space $\mathbb{H}^n$ defined as the subset of vectors of $\mathbb{R}^n$ with non-negative first coordinate (for boundary points), as seen on Figure 2.1 (right).

The *boundary* of a manifold $\mathcal{M}$ of dimensions n, noted $\partial\mathcal{M}$, is itself a manifold of dimension $n-1$ (the boundary of a surface is a curve, the boundary of a volume is a surface, etc.).

2.1.3 Vector Fields

Tangent vectors Let $\mathcal{M}$ be a smooth manifold of dimension n and $\varepsilon > 0$. A differentiable function $\gamma : (-\varepsilon, \varepsilon) \to \mathcal{M}$ is called a *differential curve* in M. Suppose that $\gamma(0) = p \in \mathcal{M}$ and let $\mathcal{D}_p$ the set of functions on $\mathcal{M}$ that are differentiable at p. The *tangent vector* of the curve γ at point p is a function $\gamma'(0)$ over functions of $\mathcal{D}_p$ defined by:



(a) Tangent space

(b) Tangent Vector field

Figure 2.2: Tangent space and vector fields over a surface $\mathcal{M}$.

$$\gamma'(0) \ : \ f \mapsto \left.\frac{d(f \circ \gamma)}{dt}\right|_{t=0}.$$

$\gamma'(0)$ associates to every function f its *directional derivative* in the direction of γ. A *tangent vector* at point $p \in \mathcal{M}$ is the tangent vector of some curve γ such that $\gamma(0) = p$. The *tangent space* of $\mathcal{M}$ at point p, noted $T_p\mathcal{M}$, is the set of all possible tangent vectors. It is a vector space of dimension n (Figure 2.2a).

The *tangent bundle* of $\mathcal{M}$ is the set defined as:

$$T\mathcal{M} = \bigcup_{p \in \mathcal{M}} \{(p, x) | x \in T_p\mathcal{M}\}$$

A *tangent vector field* X on $\mathcal{M}$ is the assignment of a tangent vector $X_p \in T_p\mathcal{M}$ for every point p of $\mathcal{M}$ (Figure 2.2b). It is sometimes referred to as a *section* of the tangent bundle $T\mathcal{M}$.

Normal vectors and surface orientation Let (Ω_a, ϕ_a) and (Ω_b, ϕ_b) be two charts. The change of coordinates $\phi_b \circ \phi_a^{-1}$ is *orientation preserving* if the determinant of its Jacobian matrix (simply called its Jacobian) is positive. An atlas is *orientable* if all its transition functions are orientation preserving, and a manifold $\mathcal{M}$ is *orientable* if it admits an orientable atlas. In this work, we will always assume that this is the case for all the considered surfaces.

Given an orientable surface $\mathcal{M} \subset \mathbb{R}^3$, we can define a local orthonormal basis X_p, Y_p at every point $p \in \mathcal{M}$ in the form of two orthogonal smooth vector fields. The orientability condition is necessary to ensure that space orientation $\det(X_p, Y_p)$ has the same sign everywhere. The *normal* vector n_p at point p can then be defined as the cross product of X_p and Y_p. Its direction does not depend on the choice of X_p and Y_p. Its smoothness with relation to p is ensured by the orientability condition.

The normal space at point p, noted $N_p\mathcal{M}$, is the supplementary subspace of the tangent space $T_p\mathcal{M}$:

$$N_p\mathcal{M} \oplus T_p\mathcal{M} = \mathbb{R}^3.$$

The normal bundle $N\mathcal{M}$ can be defined in a similar way as the tangent bundle.

2.1.4 Differential Forms

Differential forms are meant to represent quantities that should be integrated over a region of a manifold to yield a real value. Their definition stems from the introduction of a bilinear product operator $\wedge$ over $\mathbb{R}^n$ called the *exterior product*, or wedge product with the following properties:

- $(u + v) \wedge w = u \wedge w + v \wedge w$ (right distributivity)
- $w \wedge (u + v) = w \wedge u + w \wedge v$ (left distributivity)
- $\forall a \in \mathbb{R}$, $(au) \wedge v = u \wedge (av) = a(u \wedge v)$ (compatibility with scalar multiplication)
- for a basis $(dx_1, ..., dx_n)$ of $\mathbb{R}^n$: $dx_i \wedge dx_j = -dx_j \wedge dx_i$ (skew-symmetric product)

A straightforward property that follows is that $dx_i \wedge dx_i = 0$ for any vector i. Given a product $dx_{i_1} \wedge ... \wedge dx_{i_k}$, its *degree* is the number of monomials present in its expression (here, k).

A *k-differential form* ω is the assignment, for all $p \in \mathbb{R}^n$, of a linear combination of all possible products of degree k:

$$\omega_p \ = \ \sum_I f_I(p) dx_{i_1} \wedge ... \wedge dx_{i_k}$$

where $f_I : \mathbb{R}^n \to \mathbb{R}$. The space $\Omega^k(\mathcal{M})$ of k-forms over a manifold $\mathcal{M}$ is a vector space. When f_I is constant over $\mathbb{R}^n$, we get the vector space of k-linear alternating forms, which is of dimension $\binom{n}{k}$.

Exterior derivative The *exterior differentiation* is a linear operator defined over the set of k-forms that outputs a $(k+1)$-form. For a k-form $\omega = f(p) dx_{i_1} \wedge ... \wedge dx_{i_k}$, it is defined as:

$$d\omega \ = \ \sum_{j=1}^{n} \frac{\partial f}{\partial x_j}(p) dx_j \wedge dx_{i_1} \wedge ... \wedge dx_{i_k}.$$

If f is a differentiable function over a manifold $\mathcal{M}$ (i.e a 0-form) then df corresponds to the *differential* of f.

From the definition, it is easy to prove that $d(d\omega) = 0$ for any k-form ω.

A k-form ω is said to be *closed* if $d\omega = 0$.

A k-form ω is *exact* if there exists a $(k-1)$-form η such that $\omega = d\eta$. Using the fact that $d \circ d = 0$, an exact form is always closed.

Examples of forms on surfaces On surfaces, differential forms can only be of degree 0 (i.e., functions on the surface), 1 or 2. A 1-form on a surface associates, to each point $(x, y) \in \mathbb{R}^2$, a quantity of form $f(x, y)dx + g(x, y)dy$. A 2-form associates a quantity of form $h(x, y)dx \wedge dy$. For instance, ω defined as:

$$\omega = \sin(y)\, dx + x^2 y\, dy$$

is a 1-form in dimension 2. Its exterior derivative $d\omega$ is the 2-form:

$$d\omega = (\cos(y) - 2xy)\, dx \wedge dy.$$

Stokes Theorem Sometimes also referred as the fundamental theorem of multivariate calculus, the (generalized) Stokes theorem relates the integral of a differential form over a domain to the integral of its exterior derivative. More formally, if Ω is an orientable manifold with boundary $\partial\Omega$ and ω a k-form, then:

$$\int_{\partial\Omega} \omega = \int_{\Omega} d\omega.$$

1-forms and gradient An exact differential form is always closed, but the converse is not true in general. In the case of 1-forms however, the *Poincaré lemma* states that on simply connected manifolds, a 1-form is closed if and only if it is exact. In other words, closed 1-form are in this case exactly the differentials of functions on $\mathcal{M}$. Going back to the definition, a 1-form ω has general expression:

$$\omega_p = \sum_{i=1}^{n} f_i(p) dx_i$$

Meaning that for each point p, ω_p is a linear form that associates, for each vector $v \in T_p\mathcal{M}$, the value:

$$\begin{aligned}\omega_p(v) \mapsto \sum_{i=1}^{n} f_i(p) dx_i(v) &= \sum_{i=1}^{n} f_i(p) v_i \\ &= \langle X_p, v \rangle \quad \text{where} \quad X_p := \sum_{i=1}^{n} f_i(p) x_i.\end{aligned}$$

When p varies over the manifold, X_p forms a tangent vector field associated to the 1-form ω. Conversely, for any vector field X_p, one can define the associated 1-form as $\omega_p : v \mapsto \langle X_p, v \rangle$. 1-forms and tangent vector fields are therefore dual representations of the same object, except that the vector expression depends on the considered inner product $\langle \cdot, \cdot \rangle$. This is why we sometimes speak of *covectors* instead of 1-forms.

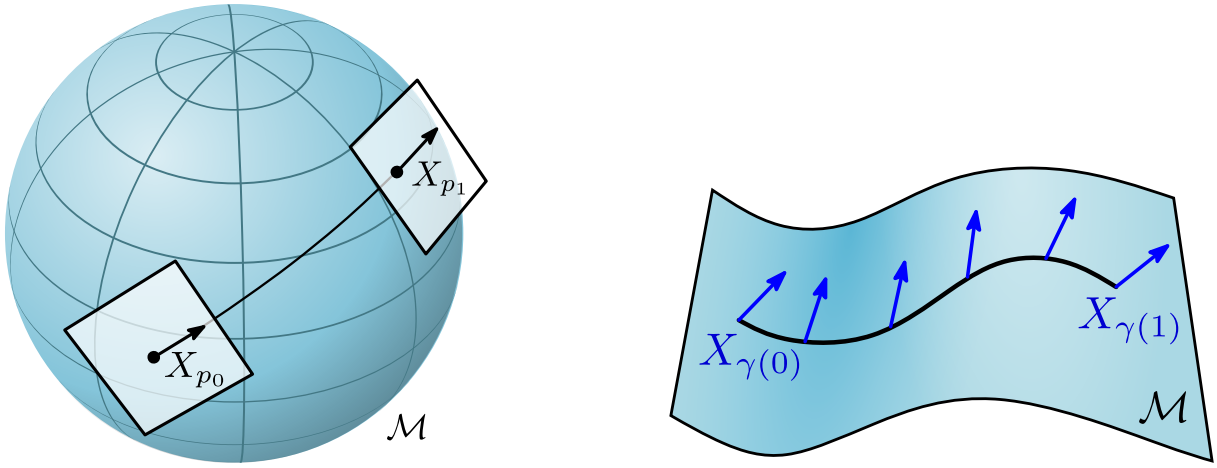
The vector X_p is called the *gradient* of $f = (f_1, ..., f_n)$ at point p, and is more often noted $\nabla_p(f)$.

2.1.5 Parallel Transport

In Euclidean space, two parallel vectors are simply collinear vectors. If their point of origin differ, one can transform one into the other by an affine translation. On a surface, the notion of parallelism is more subtle, as two tangent vectors at different point live in different tangent spaces. For example, on a sphere like Earth (Figure 2.3a), two vectors placed on different point on the equator and pointing westward can be considered parallel since they point in the same local direction, even if their orientations in $\mathbb{R}^3$ differ dramatically.

More formally, consider a tangent vector field X and an injective differentiable curve $\gamma : [0, 1] \to \mathcal{M}$ such that $\gamma(0) = p_0$ and $\gamma(1) = p_1$. X is said to be *parallel* to the curve γ at point p_0 if its *directional derivative* in the direction of γ' is zero:

$$\nabla_{\gamma'} X(p) = \left. \frac{d}{dt} X(\gamma(t)) \right|_{t=0} = 0.$$



(a) (b)

Figure 2.3: Parallel transport on a surface. (a): unlike the Euclidean case, two vectors can be parallel even if their direction is dramatically different. (b): parallel transporting a vector along the shortest path γ from $\gamma(0)$ to $\gamma(1)$ means that the angle it makes with the curve's direction γ' stays constant.

In other words, the vector field X is "constant" in the direction of the curve. This property uniquely defines a linear map $P_\gamma : T_{p_0}\mathcal{M} \to T_{p_1}\mathcal{M}$, called the *parallel transport* along γ, describing how tangent spaces and their coordinate systems relate to one another. Two vectors X_{p_0} and X_{p_1} are said to be parallel if one is the image of the other by this map.

Unlike the Euclidean case, the parallel transport between point p_0 and p_1 depends in general on the path γ taken from p_0 to p_1. If this path is the shortest path from p_0 to p_1 (a *geodesic*), then it follows that the angle X makes with the curve direction vector γ' is constant (Figure 2.3b).

Infinitesimally, a parallel transport is associated to a 1-form called a *connection* form. For a vector field X, define $\theta(p)$ as the angle between X_p and the curve's direction $\gamma'(p)$. Then, there exists a smooth 1-form ω such that:

$$\theta(p_1) = \theta(p_0) + \int_\gamma \omega.$$

2.1.6 Holonomy and Curvature

With the notion of parallel transport along a path, one natural question arises: what happens when the considered curve is closed, i.e., when $\gamma(0) = \gamma(1) = p$? We call such a curve a *cycle*. When a tangent vector X_p is transported along a cycle, we obtain a vector Y_p that is not equal to X_p in general. This phenomenon is called *holonomy*. In Figure 2.4, we can see that a tangent vector on a sphere, parallel transported around an eighth of the surface back to its original position, ends up rotated by $\pi/2$.

Let γ be a cycle that is the boundary of a topological disk $\mathcal{D} \subset \mathcal{M}$. Using the connection form ω defined above, one can express the *holonomy angle* Ψ_γ associated to γ as:

$$\Psi_\gamma = \int_\gamma \omega.$$

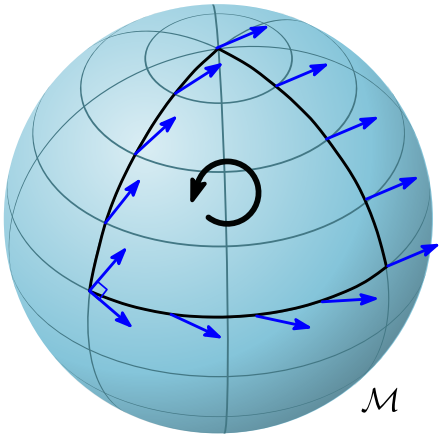


Figure 2.4: Holonomy of a vector field that is parallel transported along a cycle on a sphere.

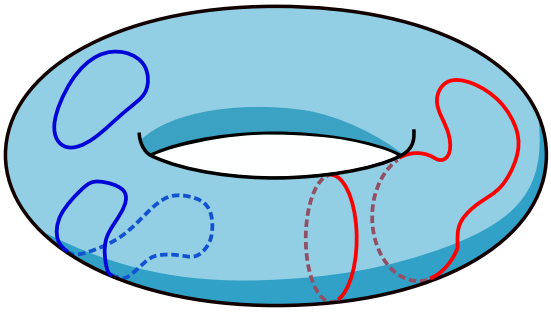


Figure 2.5: Contractible (in blue) and non-contractible (in red) cycles on a torus.

Instead of integrating along the boundary γ, we can apply Stokes theorem and integrate the exterior derivative of ω on the disk $\mathcal{D}$:

$$\Psi_\gamma = \int_{\mathcal{D}} d\omega$$

where $d\omega$ is a 2-form. By definition, it can be written as $d\omega = K dx \wedge dy$. This function K corresponds locally to a quantity called the *Gaussian curvature* up to a multiple of 2π.

To properly define the Gaussian curvature of an orientable manifold $\mathcal{M}$, one has to consider a normal vector field U. Let $p \in \mathcal{M}$ and $X \in T_p\mathcal{M}$. Define a curve γ such that $\gamma(0) = p$ and $\gamma'(0) = X$. Then, the directional derivative of U at p in direction X is the vector of the tangent plane $T_p\mathcal{M}$ defined as:

$$\nabla_X U(p) = \frac{d}{dt} U(\gamma(t))\Big|_{t=0}.$$

We define the *shape operator* at point p as its opposite:

$$\begin{aligned} S_p : T_p\mathcal{M} &\to T_p\mathcal{M} \\ X_p &\mapsto -\nabla_{X_p} U(p) \end{aligned}$$

The shape operator describes how the normal vector varies in all directions around p. It is a linear symmetric operator, meaning that for a surface, it is described by a symmetric 2×2 matrix. According to the spectral theorem, this matrix has two real eigenvalues $\sigma_1 > \sigma_2$.

The *Gaussian curvature* at point p is defined as its determinant:

$$K := \det(S_p) = \sigma_1 \sigma_2.$$

It corresponds to the same quantity integrated in the computation of holonomy above. Intuitively, it gives some information about the local shape of the surface. As shown in Figure 2.6, the surface is locally convex or concave when $K > 0$ and a saddle point when $K < 0$. When $K = 0$, an important property is that there exists a parametrization which is locally an isometry,

meaning that it perfectly conserves lengths. Surface with zero Gaussian curvature everywhere are called *developable*.

Eigenvectors associated to σ_1 and σ_2 are called principal directions of curvature. As another consequence of the spectral theorem, they form an orthogonal basis of $T_p\mathcal{M}$. When correctly defined, the principal directions of curvature form a *frame field* of the surface. This will be useful later in Section 2.4.3 as well as in Chapter 4.

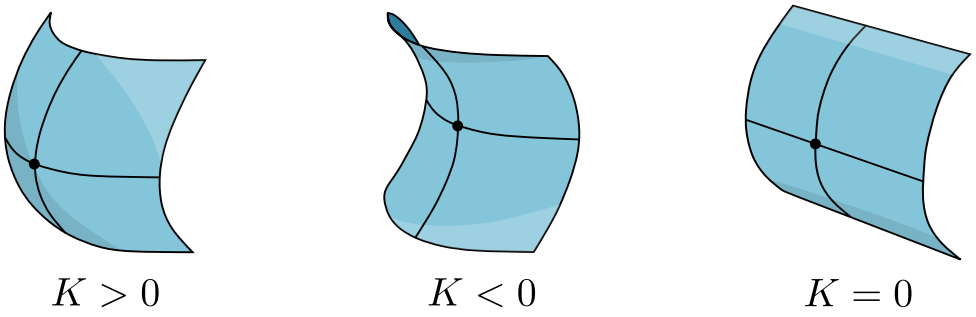


Figure 2.6: The sign of the Gaussian curvature describes the surface locally

2.1.7 Genus, Non-contractible Cycles and the Gauss-Bonnet Theorem

For the computation of the holonomy, we introduced cycles that are the boundary of a closed region of the surface. These cycles separate the surface into two distinct regions and can be continuously deformed to a single point. They are referred to as *contractible* cycles. On the other hand, there may exist a family of non-contractible cycles for which it is not the case. They typically form around handles or "holes" of the considered shape. The red cycles in Figure 2.5 are two examples of such cycles that are equivalent up to a continuous deformation.

The *genus* of a surface is an integer corresponding to its number of handles. Each handle in a manifold allows defining two additional non-contractible cycles that are not the image of each other by a homeomorphism. This means that the genus is also half the maximal cardinal of any family of non-homeomorphic non-contractible cycles.

Two orientable surfaces without boundary $\mathcal{M}_1$ and $\mathcal{M}_2$ are homeomorphic if and only if they have the same genus [O'Neill 2006, p.371].

The Gauss-Bonnet Theorem In the case of bounded orientable surfaces, the total integral of the Gaussian curvature does not depend on the geometry of the surface, but only on its genus. Let $\mathcal{M}$ be a smooth compact surface with boundary $\partial\mathcal{M}$. Just as we can define the Gaussian curvature K on $\mathcal{M}$ via the shape operator, that is the derivative of the normal vector field of $\mathcal{M}$, it is possible to define a notion of curvature on a curve like $\partial\mathcal{M}$ called the *geodesic curvature* as the angle κ between the curve's direction and the derivative of its normal. Knowing this, the Gauss-Bonnet theorem links the integral of the Gaussian curvature, the geodesic curvature of the boundary curve to a topological invariant $\chi(\mathcal{M}) \in \mathbb{Z}$ called the *Euler characteristic*:

$$\int_{\mathcal{M}} K d\mathcal{M} + \int_{\partial\mathcal{M}} \kappa ds = 2\pi\chi(\mathcal{M}).$$

While originally defined for polyhedral surfaces (see Section 2.2), the Euler characteristic can be linked to the surface's genus by the formula $\chi(\mathcal{M}) = 2 - 2g$. As a consequence, every surface

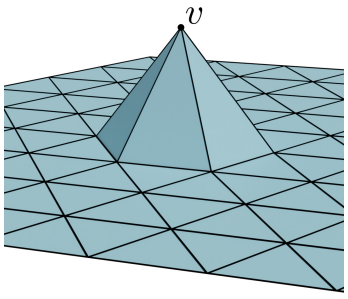
with sphere topology will have total Gaussian curvature equal to 4π ($g = 0$), any surface with a single hole, a total Gaussian curvature of 0 ($g = 1$) and so on [O'Neill 2006, p.374].

2.2 Surface Meshes

Now that the main concepts of differential geometry needed have been introduced, let us turn our attention to the way they are used in computer science. The question to ask in this section is simple : given a smooth surface $\mathcal{M}$ as before, how can we represent it inside the memory of a computer in a form that allows the design of efficient algorithms ?

Throughout the last fifty or so years, mesh structures have imposed themselves as the most widespread solution for this problem. This success can be explained by their various advantages in their two main areas of applications.

Firstly, in computer graphics, polygonal meshes, and especially triangular meshes are used to define the geometry of objects contained in a scene. Images of this scene are obtained either by computing the projection of objects as pixels in a camera's field of view (a process known as *rasterization*) or by simulating the propagation of light towards the same camera (*ray tracing*). While these methods are not limited to the use of polygonal meshes, since rendering of higher order structures like Bezier surfaces, polynomial surfaces or signed distance functions is also possible, the triangle is the simplest shape these algorithms can work on that is able to represent any geometry (as the number of triangles increases). Algorithms and their practical implementations have been highly optimized (and parallelized) for triangles in particular, leading to a situation where it might be more efficient to simply increase the number of faces in a polygonal mesh instead of switching to a more expressive representation with the same number of elements. In a context where performance is highly stressed, especially for real-time applications like video games, triangular meshes have therefore settled as the best tradeoff between computational costs, simplicity of use and quality of the final result.



The other domain in which meshes are essentials is numerical simulation, where meshes are used to represent a geometrical domain over which the solution of a partial differential equation (PDE) is computed. As such equations describe continuous phenomena, methods for discretization and approximations had to be developed. The finite element method (FEM) is perhaps the most widespread [Dhatt et al. 2012]. The key idea is to look for approximate solutions of a given PDE not in the whole intractable set of differentiable functions over the domain, but only as linear combinations of a chosen family of basis functions. Usually, polynomial functions with a single maximal value of 1 at some point of the domain are used as a basis. To better approximate the solution, one can either increase the maximal degree of said polynomials or increase the number of considered points. It turns out that in practice, the latter is preferred, as computational hardware better handles lots of simple functions rather than a small number of high degree ones [Botsch et al. 2010]. To this end, the finite element method usually uses the simplest possible functions as a basis, a so-called *hat* function (see inset Figure), that is a piecewise linear function f_v equal to 1 at some point v of the domain and equal to 0 at every other point v' such that $f_{v'}$ is also in the basis [Crane 2018]. This amounts at defining a triangular mesh of the domain whose vertices are exactly the v for which a basis function exists. Any linear combination of basis functions is then defined by its values at the vertices, which can be linearly interpolated inside triangular faces. In this framework, increasing the precision of the computation is equivalent to increasing the resolution of the triangulation.

2.2.1 Definition of a Mesh

In formal terms, a surface mesh is polygonal surface represented as a triplet $M = (V, E, F)$ where:

- V is a set of vertices, which are usually given as points in 3D space.
- $E \subset V^2$ is a set of edges, linking two vertices.
- $F \subset \mathcal{P}(V)$ is a set of polygonal faces.

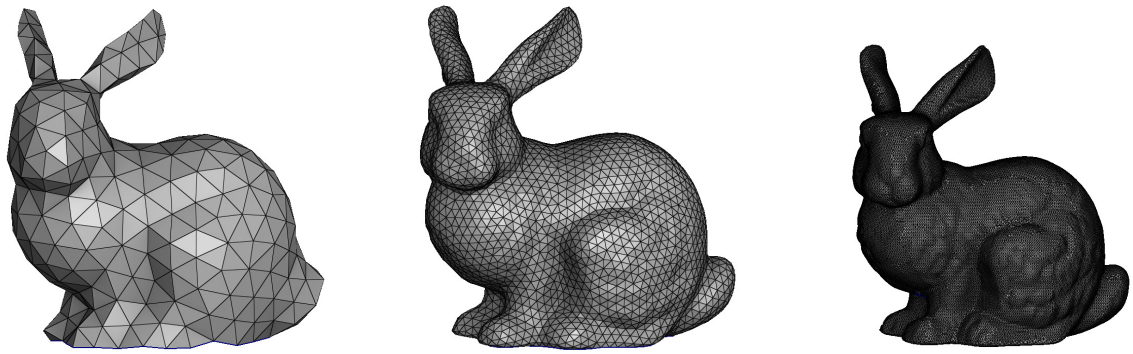


Figure 2.7: *Stanford Bunny* model as three triangle meshes with respectively 500, 5000 and 50000 triangles.

The number of faces in a mesh M and their size is the key parameter to control the quality of approximation of a smooth surface. As shown in Figure 2.7, a large number of faces allows capturing every detail of the initial model at the cost of a greater memory size and overall computation cost.

A surface mesh is *triangular* if F is only composed of triangles and *quadrilateral* if F is only composed of quadrilaterals. Triangular meshes are the most common and general structure, since every polygonal face can be triangulated. If not specified, we will always assume that a mesh is a triangular mesh. Quadrilateral meshes, which we aim at creating from triangular meshes, will be explicitly referred to as *quadmeshes*.

For a vertex $i \in V$, we will denote by $N_V(i)$ the 1-*neighborhood* of i, that is the set of vertices that are connected to i by an edge:

$$N_V(i) := \{j \in V \mid \{i, j\} \in E\}.$$

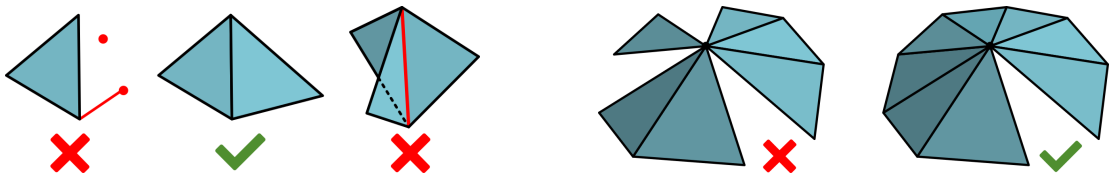
The *degree* (or valence) of vertex i is the cardinal of $N_V(i)$. Similarly, we can define $N_F(i)$ as the set of faces that admit i as a vertex:

$$N_F(i) := \{f \in F \mid v \in f\}.$$

For a surface mesh to truly discretize a smooth manifold, some additional combinatorial constraints are required on the sets V,E and F, namely:

- A vertex is always adjacent to at least one face (Figure 2.8a);
- An edge is adjacent to one or two faces (Figure 2.8a);
- $N_F(i)$ is a connected fan of triangles for any vertex i (Figure 2.8b).

A surface mesh satisfying these conditions is said to be *manifold* [Botsch et al. 2010]. This will be the case for all meshes considered in this work.



(a) A vertex is adjacent to at least one face. An edge is adjacent to one or two faces

(b) Faces indicent to a vertex should form a single connected fan.

Figure 2.8: A surface mesh is manifold if its connectivity satisfies three conditions.

2.2.2 Discrete Differential Geometry

Surface meshes can be understood as piecewise linear approximations of smooth surfaces made of polygonal patches, yet they are non-smooth by construction. This means that the definitions and theorems of Section 2.1 have to be extended or modified to account for this discrete context. As a matter of fact, it is not free to do so as some properties have to be lost in the process [Crane and Wardetzky 2017]. The field of discrete differential geometry is the study of the generalization of differential concepts and quantities over polygonal surfaces. Intuitively, the change from differential geometry to its discrete counterpart is made by not considering differential quantities anymore, but rather their integral over curves or surface patches, namely either edge paths or regions made of faces.

Tangent spaces and tangent vector fields Given a surface mesh $M = (V, E, F)$, it is possible to define a local basis X_t, Y_t of the plane defined by each face $t \in F$. The tangent space T_pM of a point inside the face is simply the linear span of the face's basis, thus the tangent space and the surface locally coincide.

Normal vectors of the mesh are defined as normals of the faces. They are not well defined on vertices and edges, but can be taken as a weighted sum of adjacent normal vectors. This allows us to also define tangent spaces at vertices as the supplementary space of the normal direction.

A tangent vector field on the mesh is the association of a vector of $\mathbb{R}^3$ to either every face or every vertex. When local bases are defined, it can be expressed as only two coordinates in this basis, sometimes grouped together as a complex number for ease of computation.

Discrete exterior calculus Just as exterior calculus defines and uses differential k-forms over smooth manifolds, its discrete counterpart was introduced in the early 2000 as a way to perform actual computations using these objects on meshes. We refer to Hirani [2003] and Desbrun et al. [2005] for a complete exposition.

As a broad overview, the theory is based on the notion of k-chains, which in 2D are the oriented edges (1-chains) and oriented faces (2-chains). Discrete k-forms are then seen as maps from k-chains to $\mathbb{R}$. That way, 0-forms correspond to functions over the mesh (defined on vertices and linearly interpolated), 1-forms associates a value per oriented edge and 2-forms, per oriented face. 1-form can be thought as the integral of a regular differential 1-form along the path that the edge defines. The same holds for a 2-form that needs to be integrated over a whole face.

Alternatively, one can consider the elements of the dual mesh as chains to define discrete forms onto. A dual edge is simply an abstract "orthogonal" edge that links two adjacent face circumcenters, and a dual face is a surface patch surrounding a vertex. In this dual context, 0-forms represent functions on faces, and 1 and 2-forms are derived in the same way. Figure 2.9 summarizes the bestiary of discrete differential forms for triangulations in 2D.

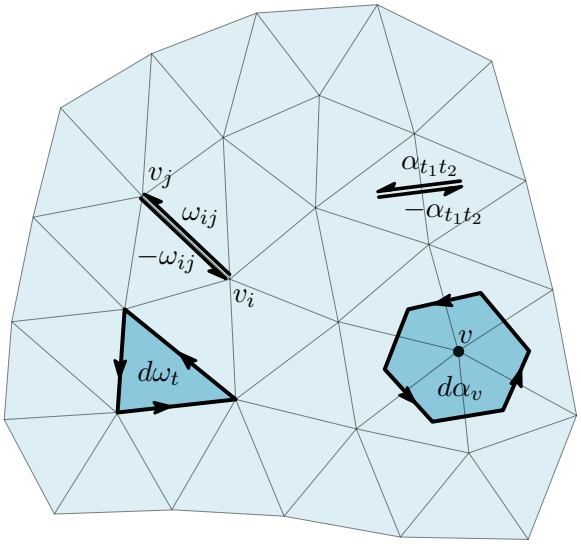


Figure 2.9: In the discrete setting, differential forms are described by values over oriented k-chains, which are primal and dual elements of the mesh. Here, ω is a primal 1-form and its exterior derivative $d\omega$ can be computed on a face t as the sum over all oriented edges around t. The same goes for the dual 1-form α, which derivative is computed per vertex v as the sum around the dual surface patch of v.

The discrete exterior derivative maps the set of k-forms to the set of $(k+1)$-forms. In practice, if α is a 0-form on vertices, $d\alpha$ is defined on edge (i, j) using Stokes theorem as:

$$d\alpha_{ij} := \int_{ij} d\alpha = \alpha_j - \alpha_i.$$

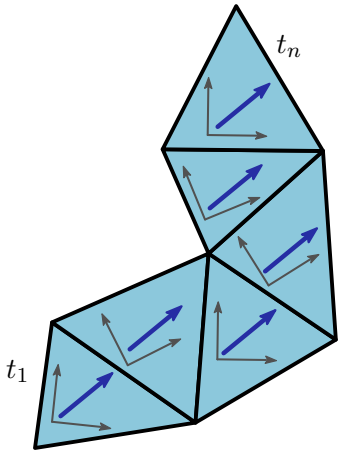
If ω is a 1-form over edges, then over every triangular face (a, b, c), $d\omega$ defines a 2-form such that :

$$d\omega_{abc} := \int_{abc} d\omega = \omega_{ab} + \omega_{bc} + \omega_{ca}.$$

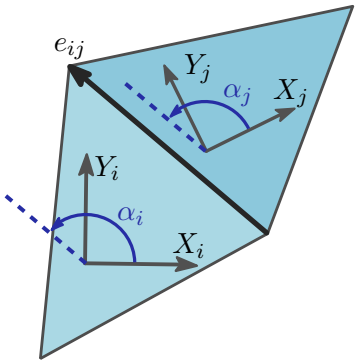
The definitions of a closed ($d\omega = 0$) and exact forms ($\omega = d\eta$) stay the same. In particular, a primal 1-form ω is exact if there exists a real function η on vertices such that $\omega_{ij} = \eta_j - \eta_i$ for all edge (i, j). It is easy to verify that $d\omega = 0$ in this case.

Discrete Parallel Transport Similarly to the continuous case, it is possible to define a parallel transport between individual faces of the mesh. Given two triangles t_1 and t_n, the equivalent notion of an injective path between them is a triangle strip $(t_1, t_2, ..., t_n)$ as the one shown in Figure 2.10a. The intuition behind the parallel transport in this case is simple: such a strip can always be flattened isometrically in the plane by "rotating" the triangles around their common edge so that their normal vectors match. After this transformation, the surface becomes euclidean and the parallel vectors are simply translations (blue vectors in Figure 2.10a). All that is left is to account for the change of local bases across edges.

More formally, we equip each triangle with an arbitrary orthonormal basis X, Y that spans its tangent space. For two adjacent triangles t_i and t_j with respective local basis (X_i, Y_i) and (X_j, Y_j) and common edge e_{ij} (Figure 2.10b), we denote by α_i (resp. α_j) the angle edge e_{ij}



(a) Transporting a vector along a triangle strip in a polygonal mesh is equivalent to flattening the strip in the plane and applying changes of local bases iteratively.



(b) A discrete parallel transport on polygonal faces is determined by a dual 1-form ρ aligning local bases

Figure 2.10: Parallel transport on polygonal meshes.

makes with the basis vector X_i (resp. X_j). Then, the parallel transport from t_i to t_j is simply a rotation of angle:

$$\rho_{ij} := \alpha_j - \alpha_i.$$

ρ defines a dual 1-form on the mesh. For our strip of triangles $(t_1, t_2, ...t_n)$, the parallel transport linear map P_{t_1,t_n} along the path can be computed as the composition of all changes of basis:

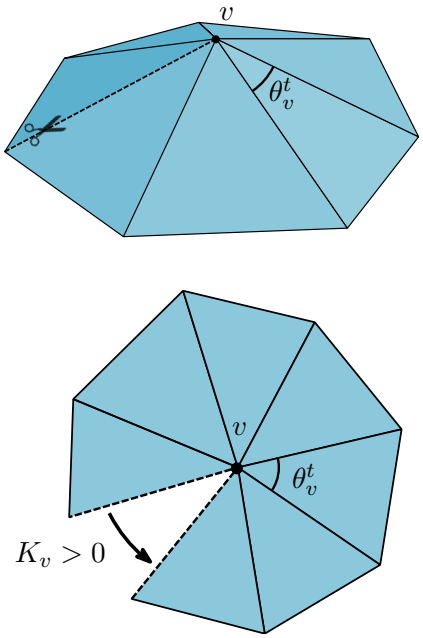
$$P_{t_1,t_n} = R(\rho_{n-1,n}) \circ ... \circ R(\rho_{1,2}).$$

Note that it is also possible to define a parallel transport between dual elements, that is to say a primal 1-form expressing rotations between vertex charts. This construction is described by Knöppel et al. [2013] and will be recalled in Chapter 4 where it will be useful.

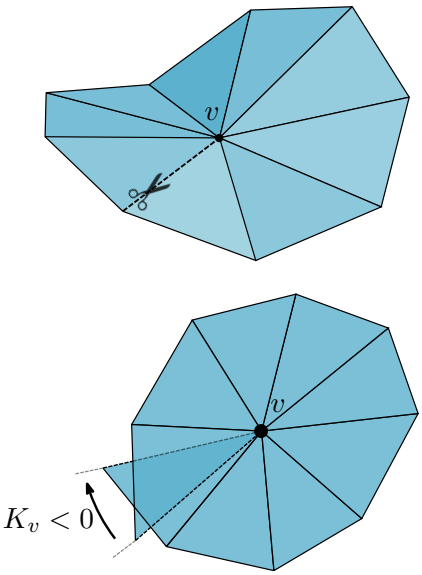
Holonomy and angle defect The discrete parallel transport also defines a notion of holonomy of the surface when we consider closed strips of triangles. In particular, consider the ring of faces adjacent to a common vertex v and a triangle strip $(t_1, ..., t_n)$ traversing all of them in order. Again, one can flatten this strip in the plane but this procedure does not imply that t_1 and t_n are adjacent (Figure 2.11). In fact, the two versions of their common edge are separated by an angle K_v called the *angle defect* of vertex v. As the name indicates, the angle defect of v quantifies the missing angle to be added around v in order for its ring to be completely flat. If we denote θ_v^t the inner angles formed at vertex v as a corner of triangle t, and:

$$\Theta_v := \sum_{t \sim v} \theta_v^t$$

the total inner angle at v, then the angle defect K_v can be computed as:



(a) Positive angle defects correspond to vertices with "pointy" neighborhoods (convex or concave)



(b) Negative angle defects correspond to saddle points on the surface (bump in some directions but recess in others).

Figure 2.11: The angle defect K_v of a vertex v is the angle needed in order to connect its triangle ring back together if we were to cut along an edge and flatten it in the plane.

$$K_v := \begin{cases} 2\pi - \Theta_v \text{ if } v \text{ is an interior vertex} \\ \pi - \Theta_v \text{ if } v \text{ is on the boundary} \end{cases} \tag{2.1}$$

The angle defect can be interpreted as a discrete version of the Gaussian curvature on vertices. As illustrated in Figure 2.11, a positive angle defect corresponds to either convex or concave neighborhoods and a negative angle defect appears in saddle-shaped regions. A vertex with a non-zero defect will be referred to as a *cone*.

Principal directions of curvature While the angle defect is simple to compute on a mesh, it is not the case for the shape operator in general. One reliable way of obtaining an approximation is described by Cohen-Steiner and Morvan [2003], where the authors define the shape operator for an edge e as the 3×3 matrix using the dihedral angled β between the normals of the two adjacent triangles:

$$S_e = \beta \frac{e.e^T}{||e||^2}$$

By averaging these matrices around the edges of a face t, one can extract a symmetric matrix S_t which eigenvalues are $0, \sigma_1$ and σ_2. Vectors in the normal direction of the considered face are always in the kernel of the matrix, while principal directions of curvature form the eigenspaces for σ_1 and σ_2.

Gauss-Bonnet theorem for polygonal surfaces More than just a generalization of the Gaussian curvature on smooth manifolds, the angle defect is the quantity needed to prove a

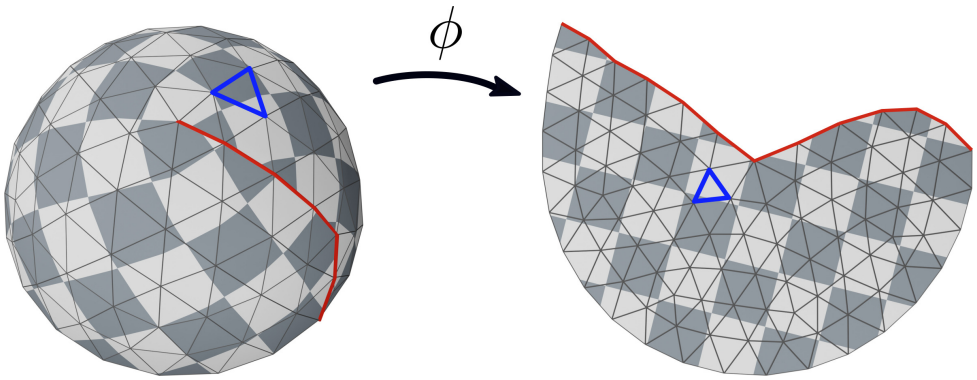


Figure 2.12: A parametrization ϕ of a surface mesh is a piecewise linear map that sends every triangle of the mesh (here, a half-sphere) to a triangle in the plane (right). Seams (edges in red) can appear when the image of adjacent triangles are not adjacent in parameter space. The uv-coordinates defined by the parametrization are represented by a checker texture

discrete version of the Gauss-Bonnet theorem. For a polygonal mesh $M = (V, E, F)$, if K_v is the angle defect at vertex v, then its sum over all vertices is a topological invariant:

$$\sum_{v \in V} K_v = 2\pi\chi(M).$$

In this setting, we also retrieve the well-known expression of the Euler characteristic in function of the number of vertices, edges and faces of a polyhedron [Euler 1758]:

$$\chi(M) = |V| - |E| + |F|.$$

2.2.3 Parametrization of a Mesh

We are now ready to properly define the problem of surface parametrization for a polygonal mesh. Going back to the definition, a parametrization ϕ associates a point in the plane to every point of a surface. Given a triangular mesh M in the numerical simulation setting described earlier, this would amount to define a function $\phi : V \to \mathbb{R}^2$ on vertices and consider its linear interpolation inside faces. But for topological reasons, we actually need to take eventual discontinuities along edges into account. Therefore, ϕ is typically described as the coordinates $\phi(c) = (u_c, v_c)$ it assigns to every triangle corner, which can vary for different corners of the same vertex. These coordinates are usually called the uv-coordinates in opposition to the xyz-coordinates of vertices in real space. In computer graphics, it is common to speak about uv-mapping instead of parametrization.

Alternatively, ϕ can be described purely as a linear mapping per triangle. For a triangle $t = (i, j, k)$ equipped with an arbitrary base, instead of describing ϕ with the 6 values $(u_i, v_i), (u_j, v_j)$ and (u_k, v_k), one can describe it as a 2×2 Jacobian matrix J_t mapping xyz-coordinates $(x_i, y_i), (x_j, y_j), (x_k, y_k)$ of i, j, k expressed in the local basis of t to uv-coordinates:

$$J_t = \begin{pmatrix} u_j - u_i & u_k - u_i \\ v_j - v_i & v_k - v_i \end{pmatrix} \begin{pmatrix} x_j - x_i & x_k - x_i \\ y_j - y_i & y_k - y_i \end{pmatrix}^{-1}.$$

The two formulations are equivalent up to a global translation. While the coordinates per corner representation is more direct to use, the matrix formulation allows for a simpler expression of the properties of the mapping during computation.

Figure 2.12 is an example parametrization of a half-sphere. Each triangle of M is mapped to a counterpart triangle in the plane. The uv-coordinates, forming a perfect grid in parameter space can be wrapped back onto the surface using the inverse of ϕ. For this to be possible, ϕ has to obviously be an invertible mapping. Recall that in the smooth context, we required the parametrizations to be homeomorphisms. Within the piecewise linear framework, we require that ϕ is *foldover-free*, meaning that is it invertible but also orientation-preserving. In practice, using the Jacobian formulation of ϕ, this boils down to constraining $\det(J_t)$ to be positive for all triangles t [Garanzha et al. 2021].

In the remaining of this chapter, we will discuss parametrization algorithms with two applications in minds. In Section 2.3, discussion about texture mapping applications will allow us to take an historical point of view on the domain and introduce its fundamental concepts. Then, Section 2.4 will focus specifically on quadmeshing. As we will see, parametrization based approaches for this task require specific constraints to be integrated into the final mapping.

2.3 Surface Parametrization for Texture Mapping

In this section, we will review parametrization algorithms from an historical perspective. We will focus on texture mapping applications since most of the early algorithms were developed with this application in mind. This will allow us to introduce the basic concepts behind surface parametrization, before moving in Section 2.4 onto our application of interest, namely seamless parametrization for quadmeshing.

The section is organized as follows. Starting from the simplest possible case of a disk topology mapped to a convex fixed domain of the plane (Subsection 2.3.1), we will progress towards more general parametrization algorithms. Subsections 2.3.2 and 2.3.3 describe two generalizations, namely free-boundary conformal maps and foldover-free maps. In Subsection 2.3.4, we discuss distortion minimizing maps with more generality and define several useful distortion metrics. Starting from Subsection 2.3.5, we will extend our review to parametrization techniques where the mapping is allowed to be discontinuous at edges, thus creating seams. This setup allows generalizing to arbitrary shapes and not only disk topologies (Subsection 2.3.6). Finally, we will discuss specific constraints to enforce onto a parametrization to make seams visually disappear (Subsection 2.3.7), hinting towards the notion of seamless parametrization.

2.3.1 Tutte's Embedding

Recall that the formal definition of a parametrization we gave at the beginning of Section 2.1 calls for a family of local charts over which coordinates are defined. Let us start by considering one of such chart C. Without loss of generality and for practical reasons, let us assume that C has disk topology. Mapping C to the plane can be done in an infinite number of ways, since every homeomorphism ϕ would represent a correct parametrization. Stated differently, C can be mapped as any shape $U \subset \mathbb{R}^2$ by a valid parametrization, as long as U also has disk topology. For simplicity, one would like some restrictions on U in order to effectively be able to compute something. A first approach, which is also the historical one, is to consider that the domain U is fixed and known in advance. In practice, C and U are both represented by two triangular meshes with the same connectivity. Assuming that U is known as a domain simply means that we know the embedding $\phi(v)$ of all vertices of the boundary $\partial U = \phi(\partial C)$. The problem is to compute the positions of every other vertices such that the mapping ϕ remains injective. In the language of vertices, edges and triangles, this simply amounts to placing all interior vertices such that no edges cross each other.

The solution to this problem was formalized in 1963 by Tutte as a general method for drawing 3-connected graphs[7] in the plane while minimizing the set of edge crossings [Tutte 1963]. In particular, Tutte described a method that finds an injective embedding of the graph as long as the boundary ∂U is convex. It simply works by assigning each interior vertex to the barycenter of its neighbors. More precisely, if we note p_i the coordinates of vertex i in the plane, Tutte's embedding can be computed by solving the following least-square problem:

$$\min_{p} \sum_{(i,j)\in E} w_{ij} ||p_i - p_j||^2. \tag{2.2}$$

under the constraint that p_i is fixed for any vertex i of the boundary ∂U.

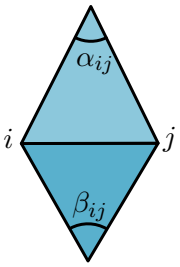
Originally, weights w_{ij} were all taken equal to 1 (barycentric embedding), but Floater [1997] later showed that the guarantee of injectivity holds for any positive weights w_{ij} per pair (p_i, p_j).

[7] A 3-connected graph is graph that remains connected whenever fewer than 3 vertices are removed. This includes polygonal meshes where each face has at most 2 vertices on the boundary.

The question of finding the optimal weights for the parametrization to be "as natural as possible", meaning that the shape and areas of triangles should be preserved as much as possible, quickly arose. In the same work, Floater derives a formula for the weights that is based on adjacent triangles' areas and amounts to consider[8]:

$$w_{ij} = \frac{1}{2}\left(\cot(\alpha_{ij}) + \cot(\beta_{ij})\right) \tag{2.3}$$

where α_{ij} and β_{ij} are the opposite angles in the two adjacent triangles of edge (i, j) (see inset Figure).



If the uv-coordinates of points $p_i = (u_i, v_i)$ are stacked as two vectors $u = (u_1, ..., u_n)^T$ and $v = (v_1, ..., v_n)^T$, Problem (2.2) can be rewritten as a matrix system:

$$\min_{(i,j)\in E} u^T L u + v^T L v \tag{2.4}$$

where L is the symmetric square matrix defined as:

$$\begin{cases} L_{ij} = -w_{ij} \text{ if } (i,j) \in E \text{ and 0 otherwise} \\ L_{ii} = \sum_{j \neq i} w_{ij} \end{cases} . \tag{2.5}$$

L is called a discrete Laplacian operator. When the considered weights are the cotangents of Equation (2.3), we obtain the *cotan Laplacian* [Solomon et al. 2014]. The cotan Laplacian is the discretization of the *Laplace-Beltrami* operator, that is to say the Laplacian matrix with the "most natural" weights that approximates at best its smooth counterpart on manifolds [Crane 2018, Ch.6]. A function u such that $Lu = 0$ is called *harmonic*. Tutte's barycentric embedding therefore amounts at finding two of such functions to be used as uv-coordinates. Figure 2.13 showcases the result of Tutte's embedding of a cow's head mesh into three different shapes, highlighting that the solution of Problem (2.4) is valid only when the target domain is convex.

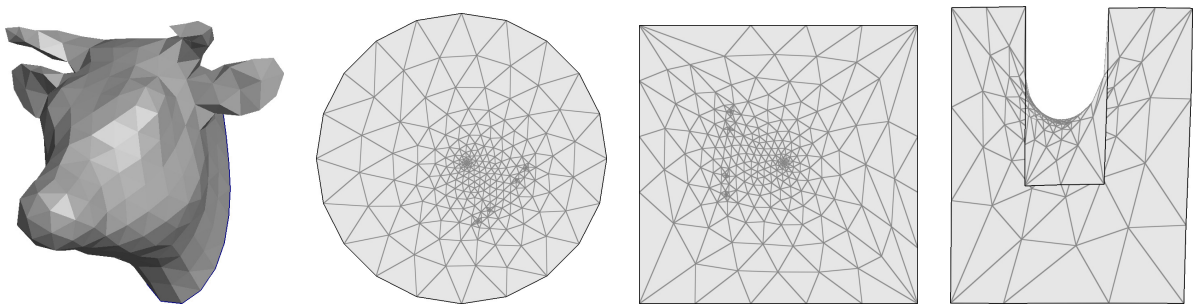


Figure 2.13: A disk-topology shape (left) is mapped into three planar shapes using Tutte's barycentric embedding. The resulting parametrization is injective as long as the considered shape is convex and weights are positive. For non-convex shapes (right), the embedding can exhibit flipped elements.

The problem of disk embedding in the plane, solved by Tutte and improved by Floater, is what sparked more general research in surface parametrization in the early 2000. Two branches of problems arose as generalizations. The first tries to parametrize a chart C without prior

[8]While these "cotan weights" are the correct discretization, they are unfortunately not always positive, which breaks the injectivity guaranty.

knowledge on the target domain's boundary (Section 2.3.2). The second attempts at generalizing the injectivity property of Tutte's embedding to non-convex arbitrary boundaries (Section 2.3.3).

2.3.2 Free-boundary Parametrization, Conformal Maps

Any embedding of the same form as Problem (2.2) needs some locked positions for vertices in order to avoid the trivial zero solution. What if we instead want a free-boundary parametrization where the target domain U is unknown ? For simplicity, the first algorithms tackling this problem restrained themselves to a specific deformation model, that is to say a subset of all possible parametrizations. A very popular deformation model is the set of *conformal maps*, for which angles are preserved. The use of this family of functions is motivated by two reasons. Firstly, the uniformization theorem states that any smooth surface with disk topology admits a conformal parametrization [de Saint-Gervais 2016], so restricting to this set of functions does not hinder the ability of any algorithm to find a solution. Secondly, a deformation $\phi(x, y) = (u(x, y), v(x, y))$ is conformal if it satisfies the linear partial differential equation of Cauchy-Riemann:

$$\begin{cases} \dfrac{\partial u}{\partial x} &= \dfrac{\partial v}{\partial y} \\ \dfrac{\partial u}{\partial y} &= -\dfrac{\partial v}{\partial x} \end{cases}$$

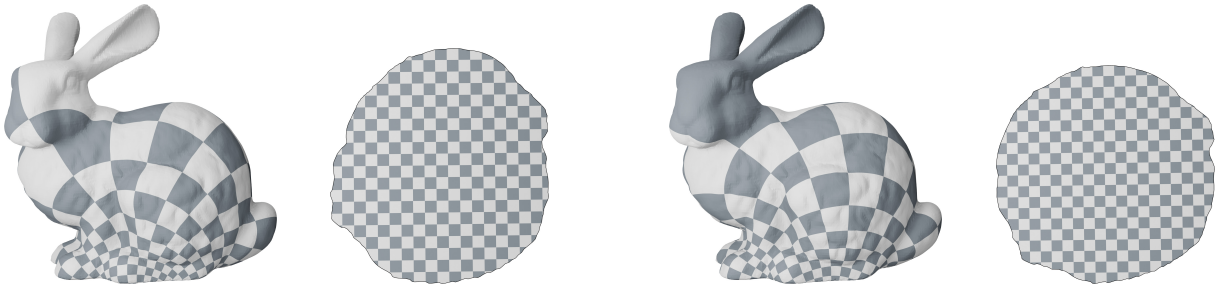
This criterion is easy to optimize over, as it boils down to solving a linear system in practice. This is the method used in the *Least-Square Conformal Maps* (LSCM) algorithm [Lévy et al. 2002]. This algorithm will be discussed in details in Chapter 3 as we expose its properties and describe an application to seamless parametrization.

An equivalent formulation of the LSCM system, called *Discrete Conformal Parametrization* (DCP) was independently developed by Desbrun et al. [2002]. Links between the two approaches hints at the relationship between conformal maps and the Laplacian operator: both Tutte's embedding and DCP aim at finding two harmonic coordinate functions, but the former avoids the trivial zero solution by fixing values on the boundary, whereas the latter instead requires two fixed points and that the gradients of coordinate functions stay orthogonal everywhere. A third approach to extract harmonic functions would be to directly compute an orthonormal basis of the cotan Laplacian's kernel using an eigenvector solver. This method was performed a few years later by Mullen et al. [2008] and leads to another equivalent parametrization method. This last approach is however more robust as it does not need any fixed points to avoid zero.

In parallel, a family of methods aimed at computing a free-boundary parametrization by directly acting on angles of the triangulation. The so-called *angle-based flattening* algorithm (ABF) [Sheffer and de Sturler 2001] computes the angles of the mesh under some constraints of validity such that the defect becomes zero everywhere except at the boundary, making the surface effectively flat. Unlike LSCM or DCP, it does not however rely on a linear least-square problem, but on a more complicated non-linear constrained optimization and therefore suffers from longer computation times. It was later improved both in terms of robustness, guarantees of injectivity as well as computation speed [Sheffer et al. 2005]. An alternative faster resolution was also proposed [Zayer et al. 2007] and presents equivalent results in terms of quality.

In Figure 2.14, we compare the result of a LSCM and a ABF approach on the Stanford Bunny model. Both methods output a very similar result where individual squares of the checker texture are scaled but never stretched (right angles are preserved), a characteristic of conformal maps.

Conformal maps have also been used for surface parametrization thanks to the relationship with the angle defect of a mesh. In the smooth setting, a conformal transformation ϕ indeed



(a) Spectral LSCM [Mullen et al. 2008] (b) Linearized ABF [Zayer et al. 2007]

Figure 2.14: Free boundary chart parametrization of the *Stanford Bunny* model with a hole at its feet, using a spectral LSCM (a) and an ABF method (b). *uv*-coordinates are represented as a checker texture onto the model (left) and as a chart in the plane (right).

corresponds to a scale at each point, meaning that there exists a function $u : \mathcal{M} \to \mathbb{R}$ called the *log scale factor* such that for all $p \in \mathcal{M}$ and $X \in T_p\mathcal{M}$:

$$\frac{||d\phi(X)||}{||X||} = e^{u(p)}.$$

Given two surface manifolds $\mathcal{M}_1$ and $\mathcal{M}_2$ with respective Gaussian curvature K_1 and K_2, the log scale factor of a conformal deformation mapping $\mathcal{M}_1$ to $\mathcal{M}_2$ is solution to the *Yamabe problem* [Aubin 1998]:

$$\Delta u = K_1 - e^{2u}K_2$$

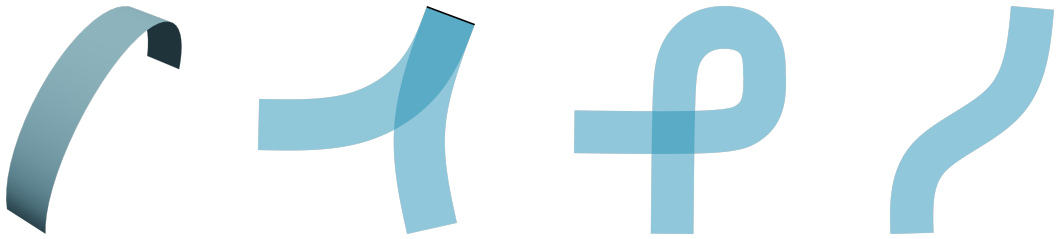
where Δ is the Laplace-Beltrami operator on $\mathcal{M}_1$. A discretization of this equation, involving the cotan Laplacian was used by Ben-Chen et al. [2008] to link a discrete scale factor on vertices to original and target angle defects K_1 and K_2:

$$Lu = K_1 - K_2. \tag{2.6}$$

When setting the target angle defect K_2 to zero everywhere, the authors can extract the scale factor that maps the initial surface to a flat one. However, solving Equation (2.6) only gives an approximation of the target metric and the flat surface needs to be recovered using other methods like ABF++ [Sheffer et al. 2005]. Alternatively, Springborn et al. [2008] minimize a convex energy whose first Newton step is Equation (2.6), which allows to directly recover edge lengths from the scale factor u and build *uv*-coordinates.

More recently, *Boundary First Flattening* [Sawhney and Crane 2018] explores the gap between fixed boundary Tutte-style methods and conformal maps. Also using Problem (2.6), the algorithm presents a way to compute a conformal parametrization with prescribed angles or lengths at the boundary, enabling extensive control over the final result. The article also discusses the impossibility of having both lengths *and* angles prescribed on the boundary while remaining conformal, highlighting the limitations of conformal maps when specific constraints need to be enforced on the mapping.

Overall, conformal maps benefit from a rich mathematical theory and enabled the design of efficient free-boundary parametrization algorithms with applications in computer graphics. Yet they remain unsuited for some applications, like quadmeshing, where necessary constraints (like distortion, injectivity or boundary control) are incompatible with an angle-preserving deformation.



(a) (b) (c) (d)

Figure 2.15: Illustration of different notions of injectivity. (a) Original object to be parametrized. (b) A parametrization that presents a foldover is not locally injective. (c) A locally injective parametrization, sometimes referred to as an immersion. (d) A (globally) injective parametrization, also called an embedding.

2.3.3 Foldover-free Maps

The other generalization of Tutte's embedding is to try to extend the same method to non-convex target domains (still with disk topology). In this broader context, harmonic coordinates functions extracted from the cotan Laplacian are no longer guaranteed to generate injective maps, as shown in the example on the right of Figure 2.13. The problem can be stated as follows: given a triangulated source domain C and a target domain U, compute a parametrization ϕ that is linear per triangle such that $\phi(\partial C) = \phi(\partial U)$ and $\det(J(\phi_t)) > 0$ for all triangle t. In other words, ϕ should conform to the boundary of the domain without creating any foldovers, that is to say triangles with negative determinants causing ϕ to locally overlap itself. As shown in Figure 2.15 this property is less restrictive than requiring that ϕ is globally not overlapping, and sufficient for many applications.

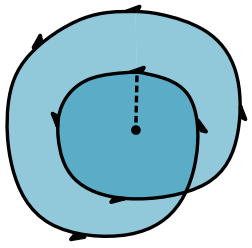
A common approach to compute a foldover-free map is to define and solve the following variational problem using numerical optimization:

$$\min \sum_{t \in T} F(J(\phi_t)) \qquad \text{where} \qquad F(J) = \begin{cases} \mathcal{D}(J) \text{ if } \det(J) > 0 \\ +\infty \text{ otherwise} \end{cases}$$

for a distortion measure $\mathcal{D} : \mathbb{R}^{2\times 2} \to \mathbb{R}^+$ (as the ones listed in Table 2.1).

As this objective is not smooth nor finite in non-injective cases, state-of-the-art methods instead define a smooth proxy function $\hat{F}$ which minimization also implies the minimization of F [Du et al. 2020; Fu and Liu 2016; Schüller et al. 2013; Xu et al. 2011]. This approach however relies on non-linear optimizers which require extensive work to prove convergence and/or correctness of the result [Garanzha et al. 2021], as well as careful tuning and design to avoid high computation costs [Claici et al. 2017].

Double coverings On top of these practical difficulties of obtaining a foldover-free map, constraining all determinants of the parametrization to be positive is sometimes not sufficient to guarantee local injectivity. A specific case can indeed appear where all triangles adjacent to a vertex are not flipped yet the neighborhood of the vertex wraps itself around two times. This phenomenon called a *double-covering* (see inset Figure) breaks local injectivity not over a face but around a vertex, and needs therefore to be avoided in



practical applications. But unlike injectivity of individual triangles, double-coverings are hard to detect and prevent in optimization since they involve several triangles which, taken independently, are all perfectly valid.

Injectivity and distortion In most works where a proxy function $\hat{F}$ is optimized to get rid of foldovers, its set of zeros is usually a strict subset of the zeros of original function F.

This is justified intuitively by the fact that a parametrization has better quality if the triangles' determinants are not only positive but "far" from zero. In other words, it is often desirable to optimize for the preservation of triangles' shape on top of injectivity, which is what most energy functions designed for the computation of foldover-free maps also do if the optimization is pushed towards its true minimum. This has also the benefit of preventing bad cases of double-coverings, which are often areas of great distortion, although no approach is able to claim any guarantee in this regard.

The problem of computing a foldover-free map is therefore closely related to the problem of distortion minimization in surface parametrization and both domains are still active areas of research.

2.3.4 Distortion Minimization

When computing a *uv*-map in computer graphics, a clear requirement imposed by artists was the minimization of the map's distortion. For texture mapping, it is indeed desirable that each pixel of the texture covers roughly the same area onto the model in order to avoid distracting visual artifacts. Moreover, avoiding shearing or stretching of the texture is also important when specific shape patterns are involved (for instance floor tiling or plaid shirt). These requirements were already appearing in the design of geographical maps, where the flat representation should reflect the reality of the curved surface of the Earth as much as possible. Preserving the shape of continents, as well as their relative sizes and positions, is therefore a natural criterion for the quality of a map.

Unfortunately, it turns out that computing a perfect map, and thus a perfect parametrization, is impossible in general. The *theorema egregium* of Gauss indeed states that the Gaussian curvature stays invariant by local isometries, that is to say transformation that preserve lengths. Since a parametrization changes the Gaussian curvature, as its maps it to a distribution where it is zero almost everywhere, it cannot be an isometry unless the considered surface already has zero Gaussian curvature. As a consequence, it is only possible to compute a parametrization that perfectly conserves angles (conformal parametrizations) or one that perfectly conserves areas (*authalic*) (see Figure 1.2 of Chapter 1) but never both at the same time. Yet, it does not prevent researchers to compute parametrizations presenting the best possible tradeoff between the two [Chien et al. 2016; Hormann and Greiner 2000; Khodakovsky et al. 2003; Lévy and Mallet 1998; Pietroni et al. 2010] for as long as the field existed.

Measuring Distortion of a Map

In order to evaluate the distortion of a parametrization, one can use many different numerical metrics. Recall that parametrizations ϕ of a triangle mesh can be described over each triangle t by a linear transformation stored as a 2×2 Jacobian matrix J_t.

Any real matrix J can be decomposed, using the QR decomposition, as the product of an orthogonal matrix Q and an upper triangular matrix R. If we consider an orientation-preserving, foldover-free parametrization ϕ, then Q is a rotation matrix and R has positive

diagonal coefficients. Since a rotation is global isometry, multiplying by Q does not introduce any distortion. We are left with:

$$R = \begin{pmatrix} \alpha & \gamma \\ 0 & \beta \end{pmatrix}, \quad \alpha, \beta > 0$$

in a given basis (e_u, e_v) of the plane fixed by the rotation Q. The values of the coefficients α, β and γ determine the nature of the transformation. A conformal transformation corresponds to the case where $\alpha = \beta$ and $\gamma = 0$. Any other transformation involves some stretching ($\alpha \neq \beta$) or some shearing ($\gamma \neq 0$), or both. However, it can be area-preserving (authalic) if $\det(R) = \alpha\beta = 1$. Only rotations ($R = I_2$) are both conformal and area-preserving.

Measuring distortion therefore involves quantifying some distance from the considered matrix to a target class of transformations. Given an orientation-preserving Jacobian matrix $J = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$, denote $\sigma_2 \geqslant \sigma_1 > 0$ its two eigenvalues. Table 2.1 defines the most commonly used metrics on J. In our work, we will be using mainly the *scale* and *stretch* distortions for quality assessment, as their optimal value correspond respectively to an area-preserving and a conformal transformation.

Metric	**Definition**	**Range**	**Optimal value**
Conformal	$\|\|J\|\|_F^2 / \det J$	$[1, +\infty)$	1
As-Rigid-As-Possible	$(\sigma_1 - 1)^2 + (\sigma_2 - 1)^2$	$[0, +\infty)$	0
Shear	$ac + bd$	$[0, +\infty)$	0
Scale	$(\det J + \det J^{-1})/2$	$(0, +\infty)$	1
Stretch	σ_1/σ_2	$(0, +\infty)$	1

Table 2.1: Distortion metrics and their definition.

Non-linear Methods for Distortion Minimization of Disk-shaped Charts

As explained in Section 2.3.3, the problem of computing low-distortion parametrizations is closely related to finding foldover-free maps. The former can be seen as an extension of the latter, where we do not only require the transformation to preserve orientation of triangle elements, but to also preserve their shape as much as possible regarding some metric.

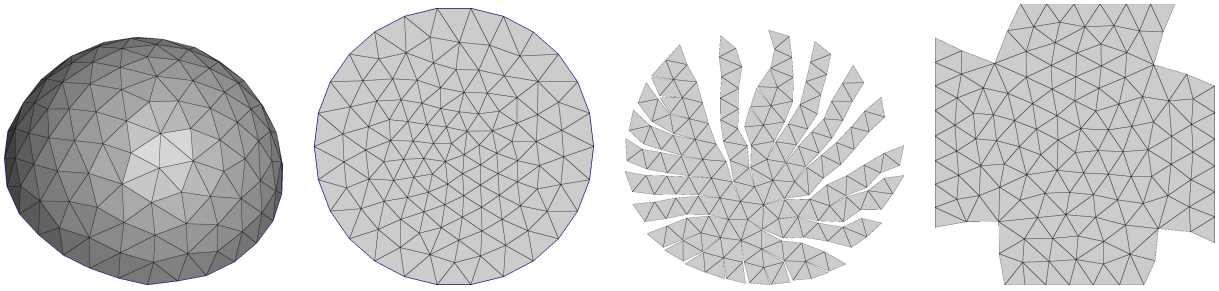
Extensive work has been done in the last decade in order to find optimal parametrizations with regard to specific metrics. Some focus on the craft of a "good" distortion metric [Levi 2023; Lipman 2012] or a good proxy with desirable properties [Garanzha et al. 2021; Rabinovich et al. 2017]. Several approaches also focused on improving the optimization process of classical distortion energies. A first one considers a sequence of intermediate results to build upon [Liu et al. 2018a]. Others consider iterative steps alternating between optimizing triangle's distortions independently and their compatibility in the final result [Liu et al. 2008; Wang et al. 2016]. Finally, some methods instead tackle the minimization problem as a whole and focus on the optimizer by using preconditioning techniques [Claici et al. 2017; Shtengel et al. 2017] which resulted in improvements in convergence and computation times.

2.3.5 Introducing Cuts

All of the methods we have mentioned so far consider a globally continuous parametrization function ϕ from a disk-topology mesh to a disk-topology shape in the plane, meaning that they restrict themselves to changes on the geometry of the model (i.e., position of the vertices in uv-space) and never change the connectivity of the input mesh. For the sake of distortion minimization, one can instead imagine a relaxation of this setup where ϕ presents some discontinuities at edges.

As a matter of fact, a perfectly isometric parametrization can always be achieved by simply laying down each triangle independently in the plane. In general however, this puzzle cannot be assembled back due to angle defect, but the parametrizations can still be made connected by stitching some edges back together along a tree, as shown in Figure 2.16c for a half-sphere model. Distortion then simply comes from the deformation needed for the parametrization to be continuous along all other edges. This is the point of view taken by the *Simplex Assembly* algorithm of Fu and Liu [2016] and an idea that we will be expanding upon in Chapter 4.

The next logical thing to look for is the best tradeoff between the number of seam edges and global distortion. This will often depend on the application and the considered model. In texture mapping where esthetic considerations play the biggest role, the number of seams is often less important than their placement into areas where they will be less visible in renderings. Algorithms that solve the problem of parametrization with cuts can be divided into two categories. Firstly, the two steps methods try to optimize only for the cuts before computing a parametrization where they are considered as boundary [Sheffer and Hart 2002; Vallet and Lévy 2009]. More recent developments instead formulate the problem as a variational objective and optimize for both the seams and the distortion at the same time [Li et al. 2018; Poranne et al. 2017; Sharp and Crane 2018; Weill–Duflos and Coeurjolly 2023]. The tradeoff between the two objectives is usually expressed as an hyperparameter which value can be tuned to get more or less seams.



(a) (b) (c) (d)

Figure 2.16: A triangulation of a half-sphere (a) can be parametrized as a disk (b) but distortion is inevitable. It can also be isometrically unfolded by stitching triangles along a tree (c) at the expense of many discontinuities. Good tradeoffs between distortion and cuts (d) can be found, for instance by using a frame field-based method (Section 2.4.3)

Interestingly, some of these algorithms work for any surface and not only disks. As soon as they are able to perform cuts of the domain, they can indeed always decompose the input shape into a subset of charts with disk topology and parametrize them independently. This paves the way for the concept of global parametrization and texture mapping for arbitrary topology.

2.3.6 Beyond Disk Topology

Going back to Section 2.1 and the continuous theory, the definition of a parametrization called for an atlas of charts over which local coordinates were defined via a homeomorphism. So far, we have reviewed many methods developed in geometry processing to compute a parametrization of such a chart. It is now time to see the bigger picture and consider parametrization of a surface with arbitrary genus.

Texture Atlases

The simplest approach to generalizing parametrization to a more complicated surface is also the most straightforward: first compute a decomposition of the surface as a set of disk-topology charts, and parametrize every chart independently. Such a decomposition can be based on greedy heuristics that encourage cuts on sharp edges [Lévy et al. 2002; Zhang et al. 2005] or iterative refinements trying to decompose the shape into charts as flat as possible [Cohen-Steiner et al. 2004]. However, such a chart decomposition algorithm would be closely related to the problem of computing the best possible set of seams on a mesh, which we have already exposed as a difficult problem to solve.

As the majority of users of texture mapping algorithms are artists applying them to animation or video games, an important feature of any decomposition algorithm lies in its interactivity. Indeed, it is still common today for artists to define their set of seams by hand. This allows hiding them efficiently depending on the context. For instance, it is common to hide the texture seams of a human face behind the ears or the head, as these regions are later hidden by the rendering of hairs. Defining by hand all the seams is still a tedious task, and algorithms were developed to instead give a good automatic first guess before allowing interactive modification [Vallet and Lévy 2009].

Deep Learning Approaches

In the last 10 years, the field of deep learning has skyrocketed and it is with no surprise that neural networks have been applied to the problem of surface parametrization. Due to their ability to approximate any continuous function [Cybenko 1989], they can be used to represent an injective mapping ϕ over a chart in a different way than the classical piecewise linear function. ϕ can be learned to map its associated chart to a canonical disk domain [Morreale et al. 2021] or map points of the plane to specific points of the surface [Groueix et al. 2018; Williams et al. 2019]. Since neural networks are differentiable functions by construction, the mappings ϕ_C over a chart decomposition can be further optimized to guarantee differentiability of the change of variables between the considered charts.

Alternatively, one could take an extrinsic point of view and represent the parametrization as a field of linear transformations applied to each triangle, and optimize the transformation over a dataset of known pairs of objects and their parametrization [Aigerman et al. 2022]. This, however, requires a classical parametrization algorithm to generate the training examples, and can only hope to generalize to other shapes with similar or worse quality.

Even if the introduction of deep learning promises interesting breakthrough in geometry processing, we are yet to see a deep learning-based method able to enforce the constraints needed for applications other than texture mapping. In particular, whether it is possible to apply a deep learning approach to the problem of quadmeshing is still an open question.

Cone Parametrization

Cone parametrization methods are perhaps the most important family of parametrization methods for arbitrary surfaces. To understand the intuition that motivates them, one can think of a parametrization firstly as a process that displaces the curvature of a surface from its original distribution to zero everywhere except on the boundary. In the case of conformal maps, this idea is expressed by Equation (2.6), which is a discretization of the Yamabe Problem. Recall that this equation links the conformal parametrization, described by a scale factor u, to the original and target Gaussian curvatures K and $\hat{K}$ by:

$$Lu = K - \hat{K}.$$

We already discussed in Section 2.3.4 how this curvature displacement is the source of the parametrization's distortion and needs to be minimal in some sense. So far, the methods we discussed only considered the field u as a variable, as K and $\hat{K}$ were completely defined. But in an effort to reduce distortion, one can instead try to find a suitable target curvature distribution $\hat{K}$ such that, for instance, the amplitude of u is minimized. This introduces specific vertices called *cones* with non-zero angle defect in the parametrization. As with the introduction of seams in Section 2.3.5, adding cones calls for a good tradeoff between their numbers and the parametrization's distortion, since setting $\hat{K} = K$ indeed leads to an isometric but highly discontinuous parametrization: the same result as Figure 2.16c.

First introduced by Kharevych et al. [2006], cone parametrization therefore aim at finding a sparse set of cone vertices that concentrate the whole curvature of the considered surface. From such a distribution, it is possible to retrieve uv-coordinates and an embedding in the plane by forcing these cones to lay on the boundary of the parametric domain. More precisely, given a set of cone vertices $S \subset V$, the following general algorithm can be applied:

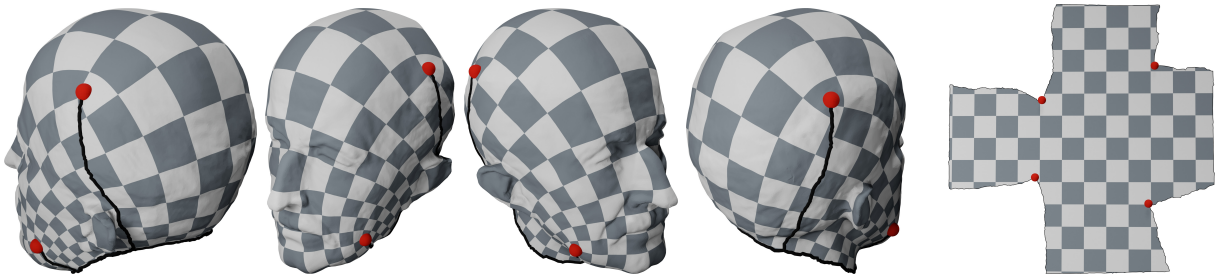
1. Perform cuts on edges in order to link every cone vertex;
2. Eventually perform additional cuts along non-contractible cycles in order to retrieve a disk topology;
3. Apply any parametrization algorithm to the cut mesh.

This procedure is illustrated in Figure 2.17. Most importantly, the cutting procedure allows the cones' triangle neighborhood to be disconnected by at least one seam edge, which means that they can be flattened in the plane as a triangle fan which angle defect is the value of the cone.

With cone parametrization, we are in a sense free from topological considerations and are able to compute parametrizations for any shape of any genus. Indeed, working with a surface of Euler characteristic χ simply means that we can add some well-placed cones of total angle defect $2\pi\chi$, cut between them and recover the disk-topology case.

On the other hand, not all cone distribution is allowed for a given surface, since the target curvature $\hat{K}$ still has to satisfy the Gauss-Bonnet theorem: the sum of its defect should be equal to $2\pi\chi$. This means that the addition of a cone with positive curvature somewhere always need to be compensated by a negative cone somewhere else.

Computing cone distributions Computing a cone distribution $\hat{K}$ that is sparse is a challenging optimization problem. Springborn et al. [2008] derive an algorithm where such cones can be prescribed and edited by hand. For an automatic placement, they rely on a greedy algorithm that concentrates local curvature to single vertices. This is also the approach taken



(a) (b)

Figure 2.17: Cone parametrization of the *Max Planck* bust model. Four points (red spheres) each concentrate a defect of $\pi/2$. Cuts are performed between them and the boundary to retrieve a disk-topology shape which is parametrized using a variant of Springborn et al. [2008]'s method (a). In the uv-domain (b), cones lay on the boundary and their neighboring triangles can be flattened as an open fan.

by Ben-Chen et al. [2008]. Both methods then rely on conformal maps to compute the resulting parametrization.

More recently, Soliman et al. [2018] propose to jointly minimize cone positions and values as well as parametrization distortion, thus computing an optimal cone distribution. In a following work, Fang et al. [2021] solve the same problem with a different optimization scheme. Overall, the sparsity condition on cones remains hard to optimize. Fortunately, we will see in Section 2.4 that efficient approaches exist when cone angles are restricted to $2\pi/n$ for some $n > 1$. This quantization idea stems from the fact that controlling the angle defect of cones allows controlling the nature of discontinuities across seams.

2.3.7 Making Seams Disappear

With the introduction of seams in a parametrization, it is possible to map any surface to a (possibly not connected) domain of the plane. However, seams are an obvious discontinuity of coordinates in the parametric domain which can form visual artifacts in the context of texture mapping (Figure 2.18 left). Fortunately, a variety of methods have been developed to control or hide the seams in order to get visually perfect results (Figure 2.18 right).

We present here two approaches to achieve this result. The first one tries to get rid of seams by mapping to domains that are not the plane but correspond to the topology of the given object. Methods from the second class instead try to restrain the discontinuities at seams to a known small family of functions which can be controlled. When using a periodic texture, one can indeed design the parametrization in such a way such that these discontinuities tear the texture in two pieces that still match one another. This is the idea between grid-preserving, or more generally *seamless* parametrizations.

Mapping to a Different Domain

As seams appear because the original surface and the target domain in the plane have different topologies, a simple solution to avoid them is to map not on the plane but onto a canonical representation of an object with the same topology as the input surface (sphere, torus, etc.). An harmonic embedding of the surface, just like the embedding of Floater [1997] for disk topology, can then be derived in the same way up to some additional topological constraints. For instance,

Figure 2.18: The introduction of seams on a mesh is sometimes necessary to compute a valid mapping but introduces visual artifacts (left). Specific constraints on the parametrization allow the seams to visually disappear when using a tileable texture (right).

Kraevoy and Sheffer [2004] develop a method for sphere-topology meshes to be embedded directly onto the canonical sphere S^2, thus generalizing the Tutte's embedding to sphere topologies. This approach is however not very useful for texture mapping, as it requires the texture to also be defined on the sphere, but has applications in shape matching and cross-parametrization.

Conformal maps have also been generalized to some extent to surfaces with arbitrary genus g. In order to compute a global conformal parametrization, Gu and Yau [2003] propose to work with its gradients, that is to say with a pair of orthogonal harmonic closed 1-forms. It turns out that the space of such forms is a linear space of dimension $2g$ closely linked to the $2g$ non-contractible cycles of the surface $(e_1, ...e_{2g})$. Their method computes a basis (ω_i) of this space such that $\int_{e_i} \omega_j = \delta_{ij}$ and then extracts its solution as a user-defined linear combination. The parametrization can then be retrieved by integration.

These two approaches do not need to define any seams onto the considered surface, but they are limited to one specific type of topology. For a way to completely hide seams that is topology agnostic, one has to rely on other means.

Towards Seamless Parametrizations

As seams are unavoidable in general, another approach trying to hide them is to work on what we can control during the parametrization process, namely the nature of the discontinuity across seam edges.

In a context of texture mapping, each point in parameter space is assigned a color via a function f, so that each point of the surface is assigned the color $f \circ \phi$. A point p on a seam will be mapped by ϕ to two distinct positions in parameter space, namely (u_1, v_1) and (u_2, v_2). A sufficient condition for the seam to be invisible is to therefore require that $f(u_1, v_1) = f(u_2, v_2)$. This is the approach taken by Ray et al. [2010], where both the mapping and the texture are optimized to get this property. This leads to a *grid-preserving* parametrization, where a discontinuity consisting of a rotation of $k\pi/2$ and an integer translation (i.e., a translation that would preserve the integer grid) would not change the color of a point.

A similar property is achieved by Aigerman and Lipman [2015] for sphere topologies without seams. The idea is to define a specific path onto the mesh that will form a virtual boundary. By virtually splitting edges along this path, the surface becomes a disk on which classical Tutte's embedding can be performed. Additional constraints on boundary edges are added, namely that they should correspond up to a known rotation of $k\pi/2$. This allows the computed chart to be grid-preserving and tile the plane. In a follow-up work, this concept was also generalized to arbitrary genuses [Aigerman and Lipman 2016].

Overall, the family of seamless parametrizations contains mapping with specific discontinuities at the seams. In texture mapping, this property can be used to design a texture image that is invariant by those discontinuities, thus eliminating any visual artifact. But this control over seams is also exactly the property needed for a parametrization to be used in quadmeshing applications.

2.4 Seamless and Grid-preserving Parametrization Applied to Quadmeshing

Now that we have concluded our overview of parametrization techniques for texture mapping, let us turn our attention to the main application we are interested in this work: quad remeshing.

This section is split into 5 parts. We start by reviewing the different quadmeshing techniques in Subsection 2.4.1. Then, we describe with more details the parametrization-based quadmeshing technique in Subsection 2.4.2. As this application requires specific constraints onto the parametrization, we give the necessary definitions and describe the usual algorithmic pipeline used to produce quads in this context. The next three subsections are dedicated to each step of this pipeline, namely the computation of a smooth frame field (Subsection 2.4.3), the computation of a seamless parametrization (Subsection 2.4.4) and its transformation into a grid-preserving parametrization (Subsection 2.4.5).

2.4.1 Overview of Quad Generation Methods

Many methods for quadmeshing have been proposed throughout the years, focusing on different aspects of the problem, like robustness, user control, quality of the elements or computation time. We present the main families of methods here, before moving on to the method of our interest, namely the parametrization-based quadmeshing pipeline. For a complete overview, we refer to the state of the art report of Bommes et al. [2013b].

Combinatorial Methods

Just like a triangle mesh can be trivially computed from a quadmesh by splitting every quads along a diagonal, a direct approach to quad remeshing consists in applying combinatorial operations to construct quads from a triangle mesh [Borouchaki and Frey 1998; Lee and Lo 1994; Rank et al. 1993].

Pairs of triangles can be evaluated according to the quality of the quad produced and merged accordingly along their common edge. In these greedy approaches, triangles that remain isolated in the end are split into three quads thanks to the addition of a central vertex and adjacent elements are refined until only quads remain (Figure 2.19a).

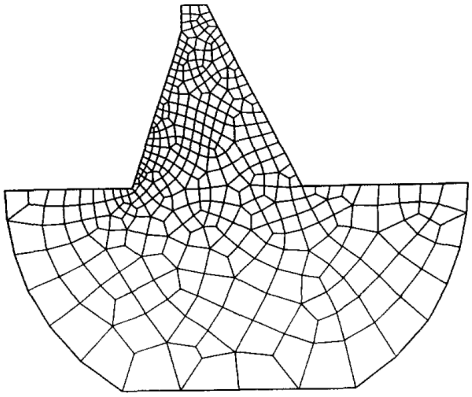
While guaranteed to produce a valid quadmesh, these methods are overall very dependent on the initial triangle layout and provide little to no control on the orientation of the final quads, on top of producing a large number of irregular vertices.

Front Propagation for Planar Surfaces

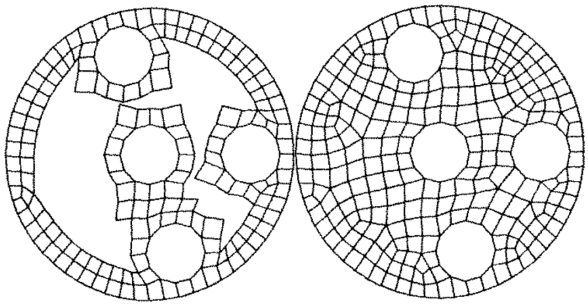
Another direct approach to quadmeshing completely discards the triangle structure and takes as a input only a boundary curve in the plane. Quad elements are then added little by little along a front, propagating towards the inside of the shape and filling the space accordingly (Figure 2.19b) [Blacker and Stephenson 1991; Remacle et al. 2013]. While robust, these methods also present many irregularities where the front meets with itself and cannot handle the meshing of curved domains.

Quadtree

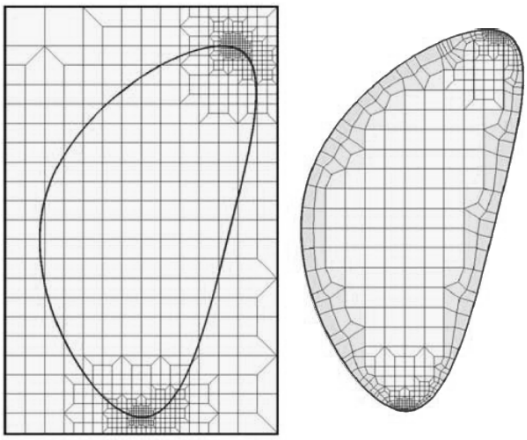
A quadtree is a tree structure that partitions 2D space in a multilevel grid. Given the contours of a shape to mesh, a quadtree can be computed by refining the grid near the boundary curve before



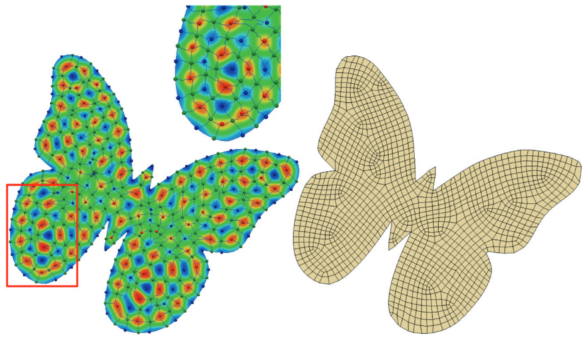
(a) Combinatorial method.
Figure from Lee and Lo [1994]



(b) Front propagation.
Figure from Blacker and Stephenson [1991]



(c) Quadtree approach.
Figure from Liang et al. [2010]



(d) Morse-Smale complex extraction.
Figure from Ling et al. [2014]

Figure 2.19: Different families of quadmeshing techniques

extracting elements from it and fixing the mesh connectivity using subdivision (Figure 2.19c).

Quadtree approaches are reliable methods to obtain a quadmesh [Jaillet and Lobos 2022; Rushdi et al. 2017]. However, making the result conform to the shape boundary usually involves irregular or poor quality elements near the boundary, where the need for good quality quad elements is often the greatest.

It is worth noting that tree approaches can be trivially generalized to the volume case of hexmeshing in 3D [Gao et al. 2019; Ito et al. 2009] for which they remain the most robust approach to this day.

Morse-Smale Approaches

Finally, a family of functions computes a quadmesh using spectral methods. Given a non-constant function over the input surface, they build its *Morse-Smale complex* (MSC), which is a graph linking its extremal and saddle points (Figure 2.19d). In two dimensions, it turns out that the MSC of any function is a quadrangulation of its domain of definition. The quality of the final quadmesh is therefore highly dependent on the choice of a function with a good oscillatory

behavior over the surface. Dong et al. [2006] proposes to use an eigenfunction f of the cotan Laplacian:

$$Lf = \lambda f$$

for some predefined value $\lambda > 0$ that will control the result's definition. Zhang et al. [2010] expands on the same idea and computes a stationary solution of the wave's equation of the form $\cos(u)\cos(v)$. These results have later been improved to support feature alignment by Ling et al. [2014].

The quality of a quadmesh produced by a Morse-Smale approach is directly determined by the considered function, over which user constraints like positions of singularities or quad element size can be hard to enforce. To alleviate these issues, Morse-Smale quadrangulations have been used alongside a parametrization algorithm in a hybrid algorithm taking advantage of both approaches [Fang et al. 2018].

2.4.2 The Parametrization-based Approach to Quadmeshing

The last great family of quadmeshing methods is based on a parametrization of the input surface. Intuitively, a quadmesh can be computed by first laying an input triangle mesh onto the plane using a parametrization technique, overlaying the uv-coordinates with an orthogonal grid of the plane and lifting the result back onto the surface. In other words, using a parametrization for quadmeshing boils down to applying a grid texture onto a surface and extracting the resulting connectivity. However, one needs to be careful of what happens at the seams and the boundary of the input mesh in order to avoid discontinuities and retrieve complete and valid quads (Figure 2.20). Just like the problem of making seams disappear in texture mapping discussed previously in Section 2.3.7, this calls for the computation of a special kind of cone parametrization with specific transition functions, namely *seamless* parametrizations.

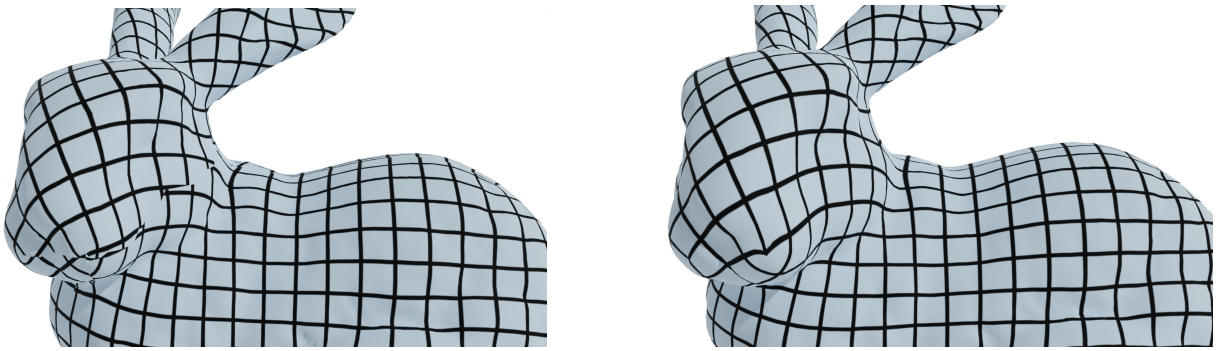
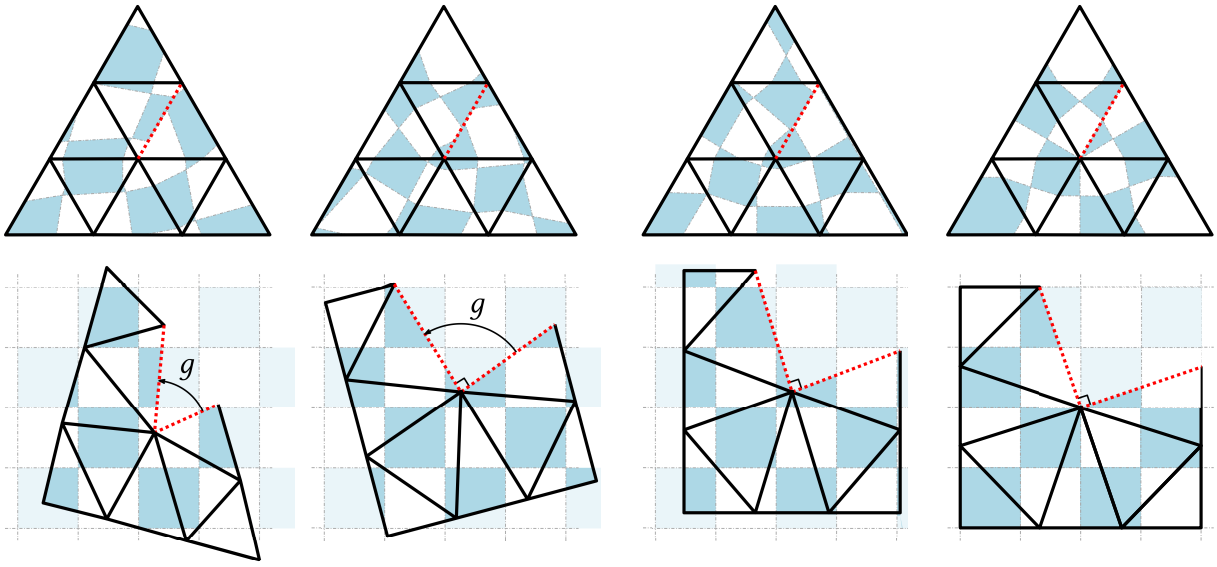


Figure 2.20: A parametrization is seamless when specific conditions on its transition functions are met. When this is the case (right), the application of the integer grid makes the seam disappear completely and yields to a valid quadmesh.

Transition Functions of a Parametrization

Let $M = (V, E, T)$ be a triangle mesh. As we already discussed in Section 2.3, a cut (or seam) in a parametrization ϕ of M occurs when two adjacent triangles in M are no longer adjacent in parameter space. As a consequence, the common edge of two such triangles is present in two



(a) Cone parametrization (b) Rotationally seamless (c) Adapted + seamless (d) Integer seamless

Figure 2.21: Four parametrization properties used in this manuscript, here illustrated by texture mapping (top) and *uv*-coordinates (bottom).

versions, one per triangle. The function g that maps one of the versions to the other is called a *transition function*.

More formally, let $(i, j) \in E$ be an edge between triangles t_1 and t_2. Points p_i and p_j are each mapped to two eventually distinct points by ϕ_{t_1} and ϕ_{t_2} in parameter space. The transition function from triangle t_1 to t_2 is defined as the map $g_{t_1 \to t_2}$ such that:

$$g_{t_1 \to t_2} \left(\phi_{t_1}(p_j) - \phi_{t_1}(p_i) \right) = \phi_{t_2}(p_j) - \phi_{t_2}(p_i).$$

By definition of ϕ, which is piecewise-linear per triangles, functions g are affine, that is the combination of a linear transformation with a translation. In other words, there exists an orientation-preserving matrix $A \in \mathbb{R}^{2\times 2}$ and a translation $\tau \in \mathbb{R}^2$ such that, for any point p:

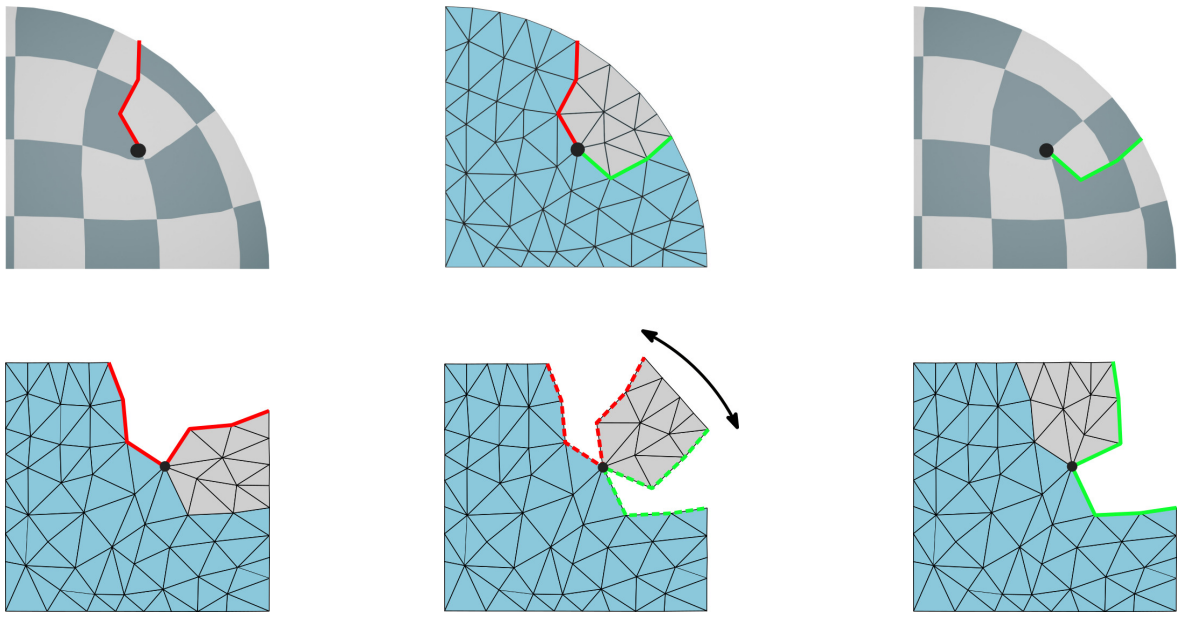
$$g_{t_1 \to t_2}(p) = Ap + \tau.$$

Given a closed loop of triangles $(t_0, t_1, \ldots, t_p, t_0)$, all incident to the same vertex $v \in V$, we can accumulate the transition functions by composition $g_v = g_{t_0 \to t_p} \circ \ldots \circ g_{t_2 \to t_1} \circ g_{t_1 \to t_0}$. The vertex v is said to be *singular* if this accumulated transition function g_v is not identity.

For a parametrization to be seamless, specific constraints have to be enforced onto the transition functions. They are summarized in Figure 2.21 and listed below.

Cone parametrization In the case of cone parametrizations, singular vertices correspond exactly to cone vertices with non-zero Gaussian curvature. Since we will always be working with cone parametrizations as of now, we will therefore use the terms "cone", "singular vertex" or "singularity" interchangeably.

A key characteristic of cone parametrizations is that since every non-singular vertex has zero Gaussian curvature, triangle rings can always be stitched back perfectly, even if they were split by one or several cuts. Lengths of different versions of a same edge should therefore match or,



(a) (b) (c)

Figure 2.22: Transition functions of a cone parametrization consist only in rotations and translations. Here, two rotationally seamless parametrizations (a) and (c) of the same quarter circle mesh make different seams. In parameter space (bottom), the discontinuity involves different edges. However, by applying the transition functions for all gray triangles (b), it is possible to transform one seam set into the other. Overall, the exact path of seams is irrelevant in the final mapping's properties. Parametrizations with this property are referred to as *global* parametrizations.

put differently, the transitions functions must consist only in rotations and translations and are free of any shearing or scaling.

Definition 2.1. A parametrization is a *cone parametrization* if for any pair of adjacent triangles $t_1, t_2 \in T$, there exists three numbers $\alpha, \tau_u, \tau_v \in \mathbb{R}$ such that:

$$g_{t_1 \to t_2} : p \mapsto R(\alpha)\, p + \begin{pmatrix} \tau_u \\ \tau_v \end{pmatrix}$$

where $R(\alpha) \in SO_2(\mathbb{R})$ is the rotation of angle α.

As a consequence, once the triangles have been independently mapped in parameter space, their relative position, defined by the position of cuts, does not matter really much: in order to move a cut, we can only apply the intermediate transition functions without introducing any distortion (Figure 2.22). In a sense, a cone parametrization is independent of the graph of cuts made on the surface. We refer to this kind of mapping as *global parametrizations*.

Rotationally Seamless A *rotationally seamless* parametrization is a special case of cone parametrization where cones can only have an integer multiples of $\pi/2$ as angle defects. This means that admissible transition functions are rotations of angle $k\pi/2$ (Figure 2.21b) plus an arbitrary translation.

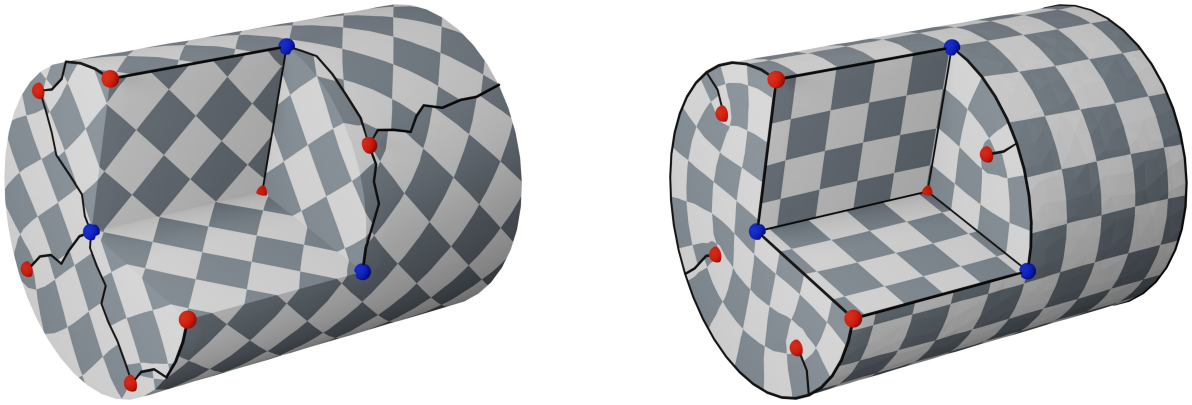


Figure 2.23: Two rotationally seamless parametrizations of a simple CAD model, obtained via our algorithm of Chapter 5. On the right, the parametrization is constrained to align with sharp edges. This constraint is necessary in practical quad meshing in order to impose quad planarity and conform to the original shape. Cones of value $\pi/2$ and $-\pi/2$ are depicted as red and blue spheres respectively. Seams and feature edges are traced in black.

Definition 2.2. A parametrization is a *rotationally seamless* if for any pair of adjacent triangles (t_1, t_2), their exists an integer $k \in \mathbb{Z}$ and two numbers $\tau_u, \tau_v \in \mathbb{R}$ such that:

$$g_{t_1 \to t_2} : p \mapsto R(\pi/2)^k \, p + \begin{pmatrix} \tau_u \\ \tau_v \end{pmatrix}.$$

More generally, methods for computing rotationally seamless parametrizations work for quantized cones of value $2k\pi/n$ for an arbitrary $n \in \mathbb{N}^+$, but the case $n = 4$ is the most studied for its application to quadmeshing and also because textures in computer graphics are naturally defined as square (or rectangular) images.

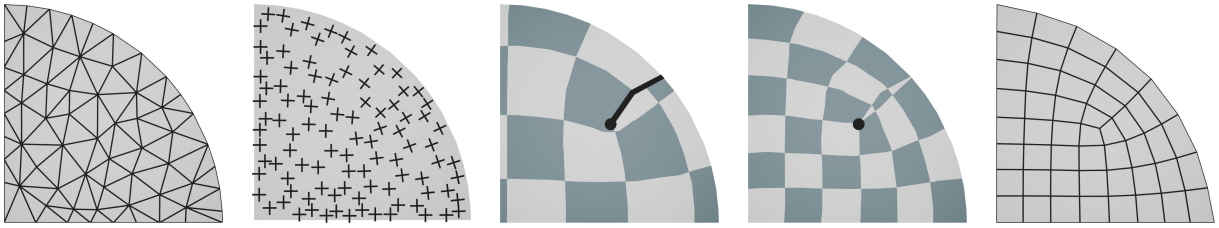
Boundary and feature adaptation A parametrization is *adapted* if all of its feature edges are axis-aligned in parameter space. This set of feature edges is usually user-defined or consists simply in a set of sharp creases or recesses. When working with CAD models, it is indeed desirable that the sharpness of these regions is conserved in the final mesh (Figure 2.23). An adapted parametrization ensures that any of these features lie on an isoline (vertical or horizontal) of the parametrization (Figure 2.21c).

Definition 2.3. A parametrization is *adapted* to its boundary (or feature lines) if all boundary edges $ij \in E$ of a boundary triangle $t \in T$ have one zero coordinate in parameter space:

$$\exists k \in \{1, 2\}, \quad \langle \phi_t(p_i) - \phi_t(p_j), e_k \rangle = 0$$

where $e_1 = (1\ 0)^T$, $e_2 = (0\ 1)^T$ are the plane axes.

Integer Seamless Finally, while a rotationally seamless parametrization limited transition functions to square-preserving rotations and arbitrary translation, a parametrization where seams can truly disappear also need to constraint the translation to only have integer coordinates (Figure 2.21d). This property is what is needed to obtain parametrizations like the one on the left of Figure 2.20.



(a) (b) (c) (d) (e)

Figure 2.24: Overview of the parametrization-based quadmeshing pipeline. Given a surface represented by a triangular mesh (a), a frame field is first computed (b) as a way to compute a cone distribution and decide feature adaptation. A seamless parametrization is then extracted (c) following the frame field's streamlines. It is further transformed into an integer seamless parametrization (d) using quantization techniques. Finally, quads are extracted (e) as isolines of the parametrization.

Definition 2.4. A parametrization is a *integer seamless* if for two adjacent triangles t_1 and t_2, there exists three integers $k, n_u, n_v \in \mathbb{Z}$ such that:

$$g_{t_1 \to t_2} : p \mapsto R(\pi/2)^k \, p + \begin{pmatrix} n_u \\ n_v \end{pmatrix}.$$

Having an integer seamless parametrization instead of just a rotationally seamless one means in particular that every cone vertex has two integer coordinates, and every vertex of a feature or boundary edge has at least one [Ebke et al. 2013]. By overlaying a regular grid onto the parameter space, vertical and horizontal lines draw a quadmesh on the surface, with points of valence different than 4 being exactly the parametrization's singularities. We also refer to integer seamless maps as grid-preserving parametrizations or integer-grid maps.

When it refers to a parametrization, the word 'seamless' alone can be ambiguous as it can both mean adapted rotationally seamless [Myles et al. 2014] or integer seamless [Fang et al. 2018] depending on the context and the algorithm discussed. In this manuscript, 'seamless' will mean 'rotationally seamless' unless stated otherwise.

The Parametrization-based Quadmeshing Pipeline

Computing a grid-preserving parametrization from which a quadmesh can be extracted is a difficult problem that is generally split into a series of more tractable steps that gradually integrate desirable properties of the final result. This algorithmic pipeline takes as an input a triangle mesh of the considered domain with an eventual description of feature edges.

1. Compute a cone distribution The first step is to decide the topology of the parametrization, namely the position and value of its cones. Recall that for the seamless property to hold, cone's defects need to be multiples of $\pi/2$. These vertices will eventually correspond to irregular vertices of the quadmesh with valence 3 and 5.

Direct approaches can be used in this step to concentrate the curvature into quantized cones [Ben-Chen et al. 2008; Levi 2021] but a popular alternative is the computation of a smooth frame field over the surface [Vaxman et al. 2016]. This latter approach not only decides completely the position of cones but also the direction of the parametrization's isolines, thus allowing easy enforcing of any feature adaptation.

2. Compute a feature adapted, rotationally seamless parametrization The next step is to compute a cone parametrization as described in Section 2.3.6, following the same cone distribution computed before. If the first step consisted in the computation of a smooth frame field, a common approach is to consider branches of the frame field as gradients of the parametrization and "integrate" them to recover uv-coordinates. This process naturally leads to a feature-adapted parametrization, allowing final quads to be guided by feature edges of the model.

3. Solve for an integer seamless parametrization Finally, the pipeline enforces the last constraint needed for a grid-preserving parametrization: that the translations across seams should be of integer coordinates. Usually referred to as the quantization step, this operation requires a mixed-integer formulation of the problem and appropriate solvers to be performed without introducing foldovers. The final integer values of translations will decide how many quad stripes are needed in each region [Bommes et al. 2009].

This pipeline is illustrated is Figure 2.24 on a simple quarter-circle model. As it turns out, splitting the problem into three or more steps did not come without drawbacks, as it makes it harder or impossible to correct suboptimal choices made previously. Whether it is possible to merge all of these steps into a single optimization will be the topic of Chapter 6.

The three remaining parts of this section will each be dedicated to an in-depth description of each step of the pipeline, namely the computation of smooth frame fields, of boundary-adapted seamless parametrization and integer seamless parametrization respectively.

2.4.3 Computing a Cone Distribution using Smooth Frame Fields

Let us start by examining the first step of the quadmeshing pipeline. The goal of this step is to compute a quantized cone distribution, that is a sparse subset of vertices with non-zero defect multiple of $\pi/2$. As direct methods for computing such a distribution have already been presented in Section 2.3.6, we focus here on the frame field method.

Let $M = (V, E, T)$ be a triangle mesh. A rotationally symmetric (RoSy) n-direction field associates, at each face $t \in T$, a set of n directions in its tangent space, which will be identified with the complex plane $\mathbb{C}$. These directions are commonly represented a unit complex numbers quantifying a rotation from an arbitrary direction $z_t \in \mathbb{C}$:

$$\left\{ \exp\left(\imath \frac{2k\pi}{n}\right) z_t \;\middle|\; k \in \{0, ..., n-1\} \right\}.$$

A key observation is that this set reduces to a single value $y_t = z_t^n$ under the application of the n-th power. In practice, such a field is therefore represented by a unique complex number per face from which all directions can be extracted [Palacios and Zhang 2007; Ray et al. 2008].

The integer n, controlling the multiplicity of the field, is unspecified in the majority of the works since algorithms working with the complex representation trick do not need to know the field's multiplicity. In practice however, $n = 1, 2, 4$ and 6 are the most common value used. In particular, a *frame field* is a rotationally symmetric direction field with $n = 4$ orthogonal directions. These RoSy-fields are part of the greater families of direction fields of surface, for which Vaxman et al. [2016] makes an extensive survey. While some generalizations, like non-orthogonal frames fields [Desobry et al. 2021; Diamanti et al. 2015], are of great interest in parametrization applications, a exhaustive review is out of the scope of this chapter.

Alignment with Feature Edges

Feature alignment can be easily captured by a frame field: for a face $t \in T$ that is adjacent to a feature edge e, it suffices to require that one the frame's direction is aligned with the edge. More formally, if $e = e_X + \imath e_Y$ in a basis (X, Y) of t, it is sufficient to set the frame representation complex as:

$$y_t = \left(\frac{e}{|e|}\right)^n \tag{2.7}$$

for one of its n-th roots to be aligned with e.

This means that frames are perfectly known around feature and boundary edges of the mesh. The goal is now to interpolate those known frames over the whole surface by "smoothing" the frame field.

Frame Field Smoothing

The use of frame fields in computer graphics dates back to the work of Hertzmann and Zorin [2000] where they were used to illustrate smooth surfaces thanks to a hatching pattern. While an arbitrary set of directions per face is a valid frame field, we are interested in practice in smooth ones, that is frame fields that minimize the difference between adjacent frames.

A first approach to compute such a field is to rely on principal curvature directions, which naturally define a frame field that is smooth enough for remeshing applications [Alliez et al. 2003]. But since estimating the curvature remains a difficult and noisy process on a mesh, a further smoothing step is often required. In practice, one would like to define some kind of distance function between adjacent frames to minimize. Hertzmann and Zorin [2000] propose to smooth the frames by representing them via an angle θ_t per face and compute:

$$\underset{\theta}{\operatorname{argmin}} - \sum_{i \sim j} w_{ij} \cos(4(\theta_j - \theta_i + \rho_{ij})) \tag{2.8}$$

where ρ_{ij} is the parallel transport 1-form over triangles defined in Section 2.2.2.

Problem (2.8) is solved using a non-linear optimizer like L-BFGS [Liu and Nocedal 1989]. When using a complex number y per frame as frame representations, it is possible to instead formulate the smoothing as a linear-least square problem [Palacios and Zhang 2007; Ray et al. 2008]:

$$\underset{y}{\operatorname{argmin}} \sum_{i \sim j} w_{ij} \, |y_j - \exp(\imath n \rho_{ij}) y_i|^2 \text{ s.t. } y_t \text{ if fixed near features} \tag{2.9}$$

This allows the problem to be more efficiently solved, with guarantees on convergence to the minimizer. However, nothing in this formulation enforces each frame to have unit norm. In practice, the presence of fixed frames near feature edges allows avoiding the trivial zero solution, but is not enough to prevent the magnitude of frames to still go to zero as we move away from these fixed values. A renormalization step is therefore necessary, but can become very noisy for low magnitude elements. To alleviate this issue, Viertel and Osting [2018] propose a diffusion-normalization scheme. Once an initial solution $y^{(0)}$ to Problem 2.9 has been computed, their algorithm alternates between a normalization of all frames and a diffusion step which combines a Laplacian smoothing and an attach term:

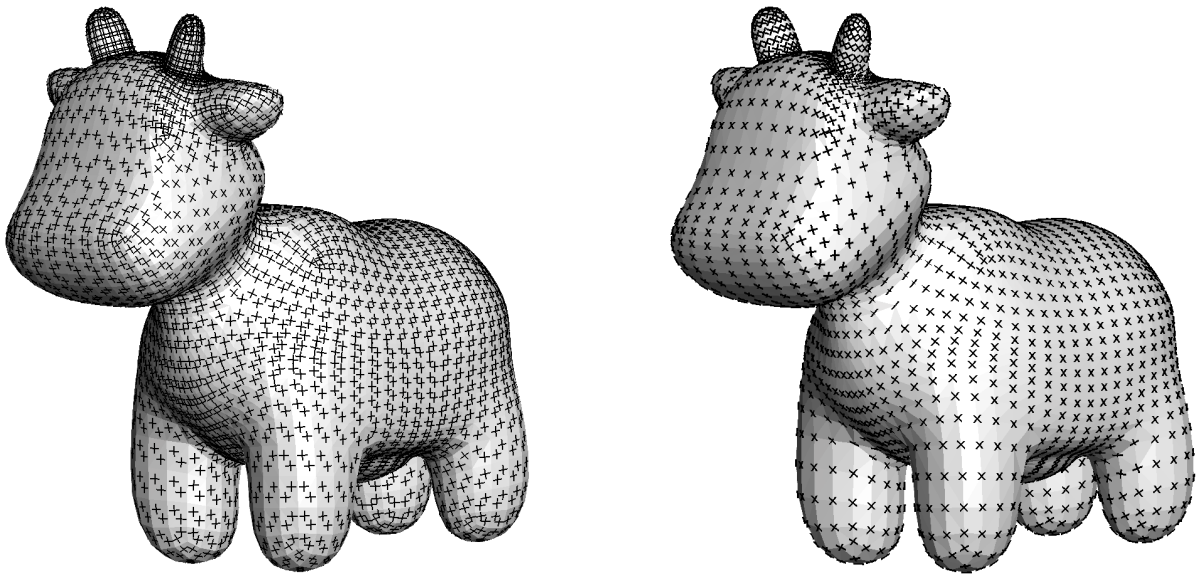


Figure 2.25: Smooth frame fields defined on the faces (left) and the vertices (right) of the *spot* model, using the diffusion-normalization scheme of Viertel and Osting [2018]

$$\left| \begin{array}{rcl} z & = & (I + \lambda L)^{-1}\, y^{(n)} \\ y_t^{(n+1)} & = & z_t / |z_t| \text{ for all } t \in T \end{array} \right. \tag{2.10}$$

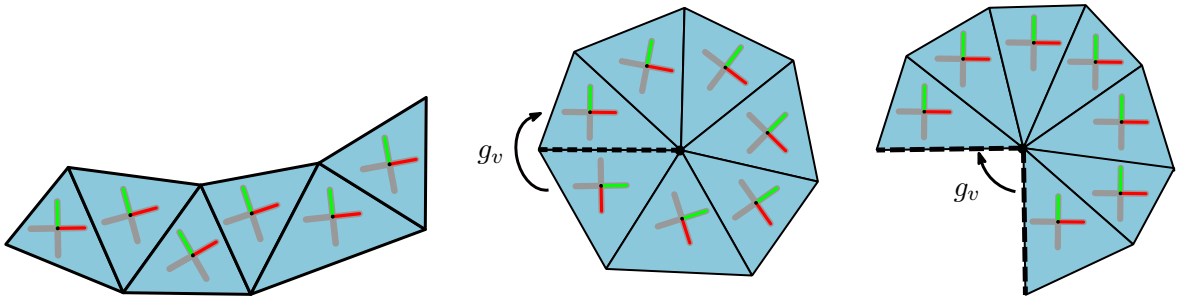
where $y^{(n)} \in \mathbb{C}^{|T|}$ are the frames at iteration n, L is a Laplacian operator (associated with weights w_{ij}) and $\lambda > 0$ is a diffusion coefficient. The algorithm performs iterations of Equation (2.10) until some convergence criterion is met. The frame fields shown on the left of Figure 2.25 was obtained after 3 iterations of this procedure.

Other methods of direction fields smoothing have been explored throughout the years. Lai et al. [2010] and Jiang et al. [2015] propose to optimize the Riemannian metric of the surface in order to accommodate various constraints while smoothing, leading to greater control on the final aspect on the field. Knöppel et al. [2013] propose a different smoothness energy that is able to control the number of singularities via a hyperparameter. Finally, some methods rely on a different representation of the frame and not a simple complex number, like spherical harmonics [Zhang et al. 2020], 2π-periodic functions [Desobry et al. 2021] or roots of a complex polynomial (PolyVector) [Diamanti et al. 2014, 2015].

Matching and Singularities

The main advantage of working with a n-direction field represented as a complex number is that one can abstract away any rotation between frames that is greater than $2\pi/n$. In more formal terms, a rotation between two adjacent frames y_i and y_j can be decomposed in two parts: first a *matching* between one direction (n-th root) z_i of y_i and the closest direction z_j in y_j, and the actual rotation that sends z_i to z_j.

This matching function g_{ij} is by construction a rotation of angle $2k\pi/n$ and can be defined for any two adjacent frames. It can be computed for a full surface mesh by traversing a tree over faces and computing g_{ij} little by little (Figure 2.26a).



(a) (b)

Figure 2.26: Matching between adjacent frames consists in pairing the closest directions of adjacent frames along a spanning tree (a). On non-traversed edges, the angle between matched directions can be greater than $\pi/2$, thus leading to a "jump" of the frame. Singular vertices happen when a loop of closest orientations around a vertex does not end as the identity (b). In that case, the composition of transition functions g_v is a rotation of $2k\pi/n$, where k/n is the index of the singularity (here, an index $1/4$).

Matching functions of a frame field share numerous similarities with the transition functions of a parametrization. Just like in Section 2.4.2, one can consider a closed loop of triangles $(t_0, t_1, \ldots, t_p, t_0)$, all incident to the same vertex $v \in V$ and compute the resulting composition of matches $g_v = g_{t_p \to t_{p-1}} \circ \ldots \circ g_{t_2 \to t_1} \circ g_{t_1 \to t_0}$. The vertex is said to be *singular* if g_v is not identity. In that case, the vertex v is said to have index k/n if g_v is a rotation of angle $2k\pi/n$, and we associate to the vertex a cone singularity of angle $2k\pi/n$ (Figure 2.26b).

Using a matching procedure, we are able to extract a cone distribution that is quantized by construction. One can show that when integrating a frame field into a parametrization (as we will discuss in Section 2.4.4), the rotationally seamless property holds, since transition functions of the parametrization correspond exactly to the matching of the frame field.

Face-based vs Vertex-based Direction Fields

So far, we have discussed frames defined on the faces of a mesh as it is the only mesh element where a tangent space can be defined without ambiguity. A "dual" discretization where frames are defined on vertices and singularities appear inside faces is also a viable alternative [Knöppel et al. 2013]. Both frame field versions share similar properties and can be smoothed and integrated using similar algorithms. The main differences will come from practical details of the targeted application. For instance, a face-based approach allows easier feature alignment whereas a vertex-based approach allows the use of the cotan Laplacian as weights w_{ij} in Problem (2.9) thus leading to smoother frame fields in general. Figure 2.25 presents an example of two fields defined respectively on the faces and vertices of the same triangle mesh. The vertex-based field is globally smoother while the face-based better follows the model's curvature at the expense of more singularities.

In order to smooth a vertex-based frame field via a solution of Problem (2.9), one needs to again define a parallel transport ρ_{ij} between adjacent frames. The procedure, described by Knöppel et al. [2013], will also be exposed in Section 4.4.2 of Chapter 4, as our representation of a frame field will be vertex-based.

Note that it is also possible to define edge-based frame fields [Wang 2006]. We refer to de Goes et al. [2016] for a comprehensive course on vector field design that goes into great details

about the subtleties of each discretization.

1-forms and Holonomy

In the continuous case of surface manifolds, we saw in Section 2.1.4 that any vector field is the dual of a differential 1-form and vice versa. This fact can be extended to surface meshes and frame fields using discrete differential calculus. For frames defined over the faces of a mesh, a 1-form ω can be defined on dual edges linking two adjacent faces. Alternatively, for frames defined on vertices, ω can be defined on primal edges. In both cases, for two adjacent frames y_i and y_j, ω_{ij} is defined by:

$$y_j = \exp(n\imath(\omega_{ij} + \rho_{ij}))y_i.$$

Intuitively, ω represents the angle of which the frame field was rotated when transported from i to j. It contains the same information as the frame y. In particular, singular vertices can be found by summing ω for all edges adjacent to a vertex. If we denote $(t_0, t_1, \ldots, t_p, t_0)$ the triangles adjacent to a vertex v_i in order, we can compute the holonomy of the 1-form as:

$$K_i = \sum_k \omega_{t_k t_{k+1}} + \rho_{t_k t_{k+1}}. \tag{2.11}$$

Just as in the smooth setting (Section 2.1.6), the holonomy of a 1-form is linked to the Gaussian curvature. As summing along a closed loop corresponds, in discrete exterior calculus, as the exterior derivative, Equation (2.11) can be more concisely expressed as:

$$d\omega = K - \hat{K}$$

where $\hat{K}$ is holonomy of the connection form ρ.

This relation, combined with the Gauss-Bonnet theorem, specifies a necessary condition on any singularity distribution: since the Gaussian curvature can be recovered from the sum of singularity indices over the whole surface, it is necessary that this sum should equal the total curvature of the surface $2\pi\chi(M)$.

1-form optimization The 1-form of view on direction fields was formalized by Crane et al. [2010] to compute a smooth frame field interpolating a set of user-defined singularities $\hat{K}$, all integer multiples of $2\pi/n$ and satisfying the Gauss-bonnet theorem. Their algorithm solves for a 1-form ω that is compatible with $\hat{K}$ in the form of a linear least-square problem with linear constraints:

$$\underset{\omega}{\operatorname{argmin}} ||\omega||^2 \quad \text{s.t.} \quad d\omega = K - \hat{K}.$$

The frame field is then recovered by fixing a random frame to a given value and propagating the relation $y_j = \exp(4\imath(\omega_{ij} + \rho_{ij}))y_i$ along a spanning tree. The fact that $d\omega = K - \hat{K}$ ensures that this procedure is independent from the choice of the tree, since eventual jumps will be multiples of $2\pi/n$ which is absorbed by the frame symmetries.

Having this 1-form point of view on frame fields also allows controlling the final position of singularities. For instance, Desobry et al. [2022] describe an algorithm where $d\omega$ is forced to be zero near feature and boundary edges, which pushes singularities away from very sharp regions of the surface, leading to nicer and less distorted results.

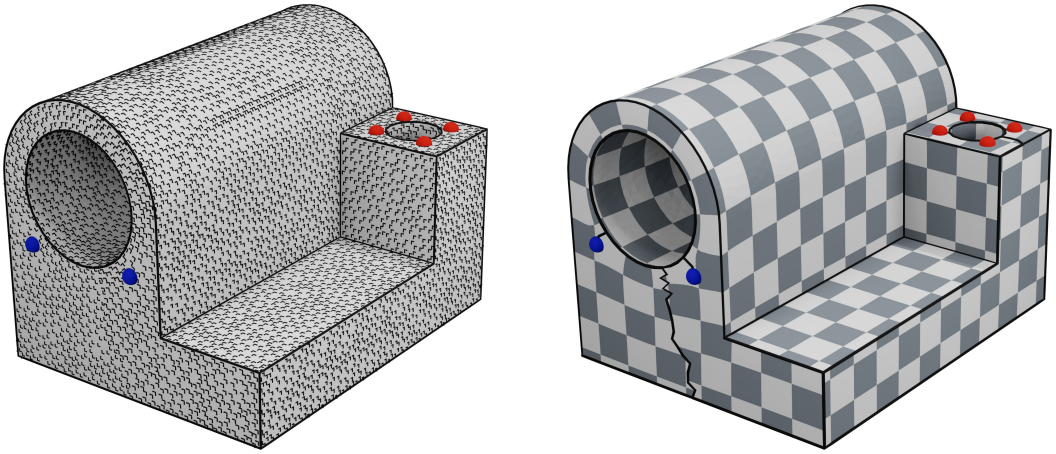


Figure 2.27: A parametrization (right) can be extracted from the matching of a frame field (left) via solving a simple linear least-square problem under linear constraints. The parametrization is adapted to feature edges (bold black lines on the left) and rotationally seamless. Singularity cones are depicted in red and blue for respectively $+\pi/2$ and $-\pi/2$ defect.

2.4.4 Rotationally Seamless Parametrization

The second step of the pipeline is the computation of a rotationally seamless parametrization. Two families of methods need to be distinguished here. Methods from the first one take as an input and a constraint a cone distribution to follow and compute a parametrization conforming to this fixed topology. This is for instance the case of frame field-based methods and some algorithms based on conformal maps. On the other hand, the second family contains methods that do not rely on such a cone distribution and instead compute one during optimization, either by a greedy approach or by restricting the set of reachable distributions.

Frame field-guided Seamless Parametrization

Firstly, a seamless parametrization can be computed from a smooth frame field. Since such a frame field defines an orthogonal frame at each point, one can indeed see the frames as a parametrization where we only allowed rotations of angle smaller than $\pi/2$ as Jacobian matrices. In that sense, the use of a frame field abstracts away any geometrical considerations about the parametrization and focuses only on its "topology", that is the position and indices of the singularity cones. In a second step, the geometry can be recovered by computing local deformations that are as small as possible. This is the approach of the *QuadCover* algorithm of Kälberer et al. [2007].

Given a smooth frame field y, *QuadCover* first computes a matching of the frames. The goal is to discriminate, among the four possible directions, two orthogonal vector fields z_u and $z_v = \imath z_u$ that will correspond to uv-coordinates of the resulting parametrization. The principle of a field-guided parametrization is to compute coordinate functions u and $v : M \to \mathbb{R}^2$, piecewise-linear per triangles, such that their gradients align at best with z_u and z_v. Unlike the original *QuadCover* algorithm and some of its refinements [Bommes et al. 2009; Myles et al. 2014], we equivalently formulate the problem in terms of the parametrization's Jacobian matrices, as it is more inline with notations of the upcoming chapters.

From the matching, a single vector field $z_t = \exp(\imath\theta_t)$ is extracted. z is smooth over the chosen tree of edges and present eventual jumps of $k_e\pi/2$ over other edges. It will be interpreted

as a rotation matrix $R(\theta_t)$. We are then interested in finding a set of jacobian matrices $(J_t)_{t\in T}$ on which we impose adaptation to feature edges and satisfaction of the transitions functions while trying to be as close as possible to the "isometric approximation" $R(\theta_t)$ given by the frame field. This boils down to the following linear least-square minimization problem with linear constraints:

$$\begin{array}{rll} \underset{J_t}{\text{argmin}} & \displaystyle\sum_{t\in T} w_t ||J_t - R(\theta_t)||_F^2 & \\ \text{subject to} & J_i e_i - R(k_e\pi/2)J_j e_j & = 0 \quad \text{for all edge } e_{ij} \text{ with jump } k_e \\ & \langle J_t e, n_t \rangle & = 0 \quad \text{for all feature edge } e \end{array} \tag{2.12}$$

The least-square objective, taken as the Frobenius norm of the difference between our matrix variables and the field's rotations, expresses the fact that we are looking for a parametrization that matches the frames at best. The first constraint acts on jumps: the two images of an edge e with adjacent triangles t_i and t_j, namely $J_i e_i$ and $J_j e_j$ (e_i are e_j being coordinates of e in arbitrary but fixed local bases of t_i and t_j) should match up to a rotation of $k_e\pi/2$. The second constraint imposes feature edge adaptation and simply states that the image of a feature edge e should either be vertical or horizontal, depending on the dot product between e and z_t. This is expressed via the basis vector $n_t = (1\ 0)^T$ or $(0\ 1)^T$.

Given a field of Jacobian matrices $(J_t)_{t\in T}$, uv-coordinates can be simply recovered by applying the matrices to original triangles of the mesh and stitching everything together along the same tree used for the matching. Figure 2.27 illustrates the recovered parametrization next to the original matching of the frames.

Integrable Frames Fields

Given a smooth frame field, a theoretical question remains: is it always possible to retrieve a global parametrization that is foldover-free, seamless and aligns with the field ? The short answer is no. In Problem (2.12), there is indeed absolutely no guarantee that resulting Jacobians have positive determinants. As such, even a seemingly good frame field can lead to degenerate or flipped triangles in the final result. An example of such a case is given in Figure 2.28. Usually, these situations can be fixed by adding a pair (dipole) of singularities with opposite indices, hence the solution wanted is no longer the smoothest field but something else. This approach for frame field repairing was developed by Diamanti et al. [2015]: representing the frames as polyvectors, that is roots of complex polynomials, their method is able to optimize over an integrability constraint that guarantees the validity of the extracted parametrization provided convergence. As a result of this optimization, singularities are added when needed and the algorithm is effectively able to solve situations like Figure 2.28.

The problem of designing frame fields that can be integrated into a parametrization can also be approached by relaxing the definition of a frame field and instead consider pairs of vector fields z_u and z_v whose norms can vary independently or that are allowed to non longer be orthogonal. These representation have been investigated as a good compromise between rigid frame fields and the full Jacobian matrix representation of a parametrization, calling either for a different representation of the field [Desobry et al. 2021; Diamanti et al. 2014] or a change of the surface's metric [Jiang et al. 2015; Lai et al. 2010].

When no feature alignment is required, sufficient conditions for a frame field to yield a valid parametrization have been later made explicit by Campen et al. [2019] and Shen et al. [2022]. In a few words, except for the specific case of two singularity cones on a zero genus surface, these works prove that a valid parametrization can be extracted from any frame field whose indices satisfy the Gauss-Bonnet theorem.

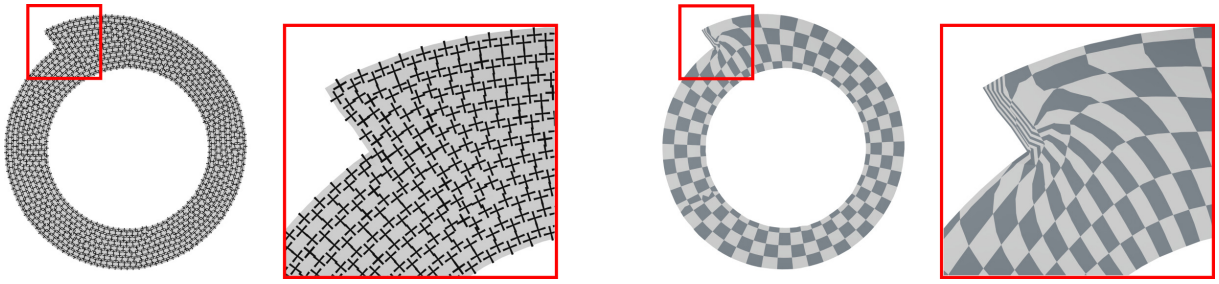


Figure 2.28: All frame fields cannot be successfully integrated into a seamless parametrization that is injective. On this model, the smoothing algorithm outputs a boundary adapted frame field with no singularity cones (left). Integrating this field into a parametrization crushes the outward step, leading to degenerate triangles.

However, finding necessary integrability conditions on a frame field in the presence of feature alignment constraints remains an open problem to this day. In Chapter 4, we will propose an algorithm that provably provides a valid seamless parametrization in this case, provided convergence of a non-linear optimization.

Direct Methods for Seamless Parametrization

The alternative to frame field-based parametrizations is to directly combine the computation of a cone distribution and the associated mapping in a single optimization. Myles and Zorin [2012] propose for instance to first find cone positions using a greedy curvature concentration approach, and then perform iterative rounding of cones' defects in order to converge to multiples of $\pi/2$. The final parametrization is then recovered by minimizing the ARAP distortion [Liu et al. 2008, and Table 2.1]. In a follow-up work, Myles and Zorin [2013] are able to integrate feature edge adaptation constraints using a method where a conformal scale factor is computed along a tree starting from features. They also demonstrate experimentally that even in the case of quantized cone singularities, adding more cone leads to less distortion.

More recently, Levi [2021] proposes a direct algorithm for computing a seamless parametrization with bounded maximum distortion. The method relies on a tree along which triangles are flattened, and cones are added when enough angle defect is accumulated.

In the case where the cone distribution is already known (for instance, emanating from a frame field), Campen et al. [2019] show how to extract a parametrization with the same topology using the framework of conformal maps.

Polycube Maps

Finally, polycubes represent a last family of seamless parametrization methods. In short, a polycube-map is a deformation of a mesh that aligns every face with one of the six normal directions defined by the unit cube. Introduced by Tarini et al. [2004], this mapping naturally defines a seamless parametrization where singularities correspond to corners and seams run along cubes' edges. This amounts to restraining to a specific set of cone distribution for which the surface can still be embedded in $\mathbb{R}^3$. However, computing a valid polycube flagging, that is a assignment of normals to every face that does not involve a foldover, is a challenging problem that is still open to this day.

The polycube approach is more widely used in the domain of hexmeshing as it can be trivially generalized to volume meshes.

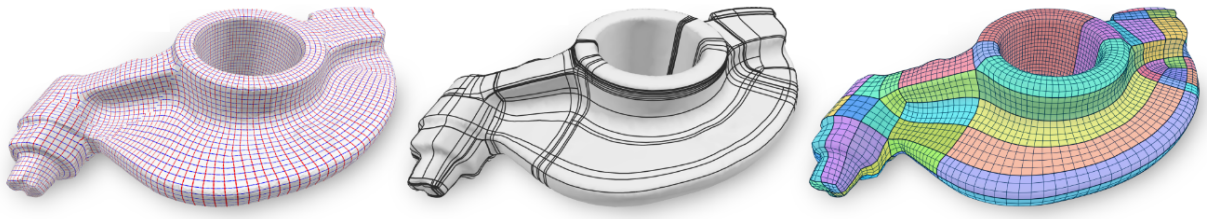


Figure 2.29: Parametrization quantization using a T-mesh. A rotationally seamless parametrization (left) is decomposed by a set of axis-aligned lines in parameter space to form a T-mesh on the surface (middle). The position of these lines is then optimized using a mixed-integer solver to recover a valid quad layout (right). Figure from Lyon et al. [2021].

2.4.5 Grid-preserving Parametrization

To conclude this section, let us discuss the third condition to be imposed on a parametrization for quadmeshing. Recall that a quadmesh can only be extracted from a parametrization whose transition functions are rotations of $k\pi/2$ and *integer* translations. The quantization of rotations has been addressed with all the methods cited previously. For the translation part, two families of methods have, again, emerged. The first one uses integer linear programming solvers to transform a rotationally seamless parametrization into a grid-preserving one. The second is closer to Morse-Smale approach for quadmeshing and utilizes a change of variables to directly compute a grid-preserving parametrization from an input frame field.

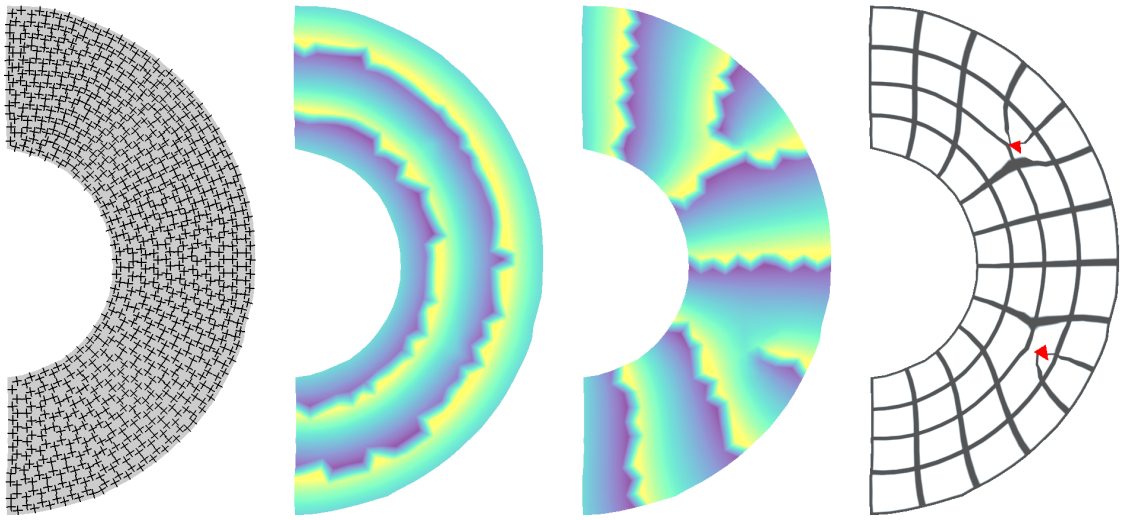
Mixed-integer Approaches

Imposing translation part of transition functions to be of integer coordinates is a problem that is different in nature than the rotation part. As the latter can be solved using simple linear solvers with objects like frame fields, the former requires to work with integer variables one way or another, which usually involves solving NP-hard problems.

Recall that a parametrization is grid-preserving if, equivalently, all its singularities have two integer coordinates and all of its feature vertices have at least one. The naive approach to imposing this constraint is to snap each vertex to the nearest point of the integer grid. However, this procedure quickly introduces foldovers or degeneracies in the parametrization which are then impossible to recover. A more careful approach is to add injectivity as constraint and formulate everything as a large mixed-integer problem [Bommes et al. 2013a, 2009]. In a effort to reduce the number of variables and constraints in instances of such problems, quantization methods sometimes rely on the construction of a T-mesh (Figure 2.29), that is a decomposition of the surface into large axis-aligned quadrilateral regions on which the integer constraints can be enforced [Campen et al. 2015; Lyon et al. 2021], or the decimation of the input [Coudert-Osmont et al. 2023].

Periodic Global Parametrization

The *Periodic Global Parametrization* (PGP) algorithm of Ray et al. [2006] avoids formulating the computation of an integer seamless parametrization as a mixed-integer problem and instead rely on a continuous optimization over a set of indirect variables. The key idea relies on the following observation: if coordinates function u and v of a parametrization are integer seamless, then the function:



(a) (b) (c) (d)

Figure 2.30: Given an input frame field (a), PGP-based methods are able to enforce integer seamless constraints without the need for a mixed-integer formulation. Two smooth periodic functions are optimized with compatible boundary and seam constraints (b and c), from which uv-coordinates are recovered. However, a full integer seamless parametrization cannot always be extracted due to the apparition of T-junctions (d)

$$\theta : (u, v) \mapsto (\cos(2\pi u), \sin(2\pi u), \cos(2\pi v), \sin(2\pi v)) \in \mathbb{R}^4$$

absorbs any integer translation thanks to the trigonometric function's periodicity and the square-preserving rotations only cause a known permutation of the four components. Given a smooth frame field with computed matching, jump functions and seams, *PGP* computes the smoothest function θ via a linear system involving the cotan laplacian matrix, and then recovers the best corresponding (u, v) (Figure 2.30). Similar approaches were derived afterward, including the works of Knöppel et al. [2015] which formulates the problem in terms of complex numbers, and Fang et al. [2018] which hybridizes the approach with a more traditional frame-field integration algorithm as a way to repair faulty areas. These methods do not strictly compute integer seamless maps as they can exhibit broken triangles (so-called T-junctions) in the parametrization, that is triangles containing two or more singularities. Nonetheless, a valid quadmesh can be reassembled from locally extracted quads.

The *PGP* algorithm and its variants will be discussed in more detail in Chapter 6, where we will discuss a method for optimizing a similar set of periodic functions without a priori knowledge on the cuts and cones positions.

Conclusion and Motivations of this Work

In this chapter, we have provided an overview of surface parametrization methods in geometry processing with applications in texture mapping and quadmeshing. In the latter case, we presented a parametrization-based quadmeshing technique where a grid-preserving parametrization is computed in several steps. However, the resulting pipeline of operations poses specific difficulties, like the integrability of frame fields, injectivity of parametrization or robustness of mixed-integer quantization to cite a few.

The goal of our work is now to propose new optimization algorithms to improve one or several steps of the quadmeshing pipeline described in Section 2.4.2. More precisely, we will try to provide an answer to the three following problems:

1. What is a "good" cone distribution for a seamless parametrization ? Even in the case of quantized cones (with value $k\pi/2$), it has been experimentally conformed that adding cones reduces the parametrization's distortion [Myles and Zorin 2013]. Yet in most state-of-the-art methods, little to no control is given over the cone distribution when computing the seamless mapping.

We will describe two new methods of computing a cone distribution and an associated seamless parametrization. The first one, described in Chapter 3 derives an energy function from the LSCM algorithm of Lévy et al. [2002] which minima correspond to flat surfaces. We extend it to the case of seamless parametrization and cone metrics on surfaces via combinatorial tricks.

In the second method, we formulate the problem of finding a cone distribution and its corresponding parametrization in a single continuous optimization problem. This allows optimizing for given properties of the final mapping (low conformal/scale distortion, feature adaptation, presence or absence of cones in user-defined areas, etc.) on *both* the mapping's Jacobian and the position of cones. The method is based on Cartan's method of moving frames and more specifically on the discretization of Cartan's two *Structure equations*. Two versions of an algorithm are proposed: the vertex-based version of Chapter 4 provably output valid seamless parametrizations provided convergence of the optimization, while the face-based version of Chapter 5 is faster and simpler to implement.

2. Is it possible to compute an *integer* seamless parametrization by minimizing a single continuous objective ? In a sense, the moving frame method of Chapters 4 and 5 unify the first two steps of the quadmeshing pipeline, namely the computation of a smooth frame field and an associated rotationally seamless parametrization. From this point, state-of-the-art methods often make use of a quantization step that relies on a mixed-integer solver. In parallel, methods based on the PGP algorithm of Ray et al. [2006] can be seen as a way to directly solve for an integer seamless parametrization given a smooth frame field as an input, thus combining two other steps of the pipeline. More interestingly, they do not use any integer in their formulation but only continuous variables optimized via least-square methods.

In Chapter 6, we will try to combine a PGP approach with our moving frame formulation in order to solve for an integer seamless map in single continuous optimization. This can be seen as both a generalization of PGP where singularities are not known in advance and a generalization of our moving frame method where we add the integer translation constraints.

3. Is it possible to formulate the seamless parametrization problem in a way that is generalizable to the volume case ? While we focus here on the surface case of quadmeshing, another related problem is the automatic generation of hexmeshes, which are volume meshes

made of hexahedral elements. As it turns out, hexmeshing is even more challenging than quadmeshing, as the transposition of the quadmeshing pipeline to three dimensions poses a whole new set of theoretical and practical difficulties. Several theoretical results do not hold anymore in three dimensions, and whole families of parametrization techniques (like for instance conformal maps) cannot be generalized. In particular, computing integrable frame fields in 3D is still an open problem.

In this context, designing parametrization algorithms that are dimension-agnostic and can also work in 3D with minimal theoretical changes is a sensible approach. It turns out that using the same exact equations as in Chapters 4 and 5, an algorithm that simultaneously computes a 3D frame field and a volume parametrization could be derived. Although it remains a promising track for future work, we will briefly describe this idea in our concluding remarks (Chapter 7).

3

Computing Cone Distributions via Continuous Optimization on a 4-covering

Contents

3.1 4-covering of a Triangle Mesh 71
3.1.1 Decomposition as triangle rings 71
3.1.2 Properties of the 4-covering 73
3.2 Flattening Energy 73
3.2.1 Least-Square Conformal Maps 74
3.2.2 Reformulation of the LSCM system 75
3.2.3 Energy Interpretation 77
3.3 Optimization of the Flattening Energy 77
3.3.1 Optimization Problem without 4-covering 78
3.3.2 Optimization Formulation over a 4-covering 80
3.3.3 Practical Algorithm 81
3.4 Results and Discussion 82
3.5 Conclusion 82

Any seamless parametrization of a surface is characterized by a set of quantized cones that determines its topology. This (usually sparse) set consists of points that concentrate the curvature of the surface in multiples of $\pi/2$. For a quadmeshing application, their distribution over the surface has a critical impact on the quality of the final mesh since it determines the position and number of its irregular (not incident to 4 quads) points in the final quadmesh.

Given an input surface, represented as a triangle mesh, finding a good quality cone distribution is a hard problem to solve due to the discrete nature of variables involved, both in position over the mesh (at vertices) and in angle defect ($k\pi/2$ with $k \in \mathbb{Z}$). Directly solving for such a distribution involves mixed-integer problems which are notoriously hard to solve in practice [Li et al. 2022]. To circumvent this difficulty, two approaches can be drawn: either rely on proxy variables like frame fields [Vaxman et al. 2016] or on greedy incremental methods [Levi 2021; Myles and Zorin 2012]. Yet, no state-of-the-art approach is able to compute a cone distribution in order to accommodate for some given quality criterion over the final parametrization, such as

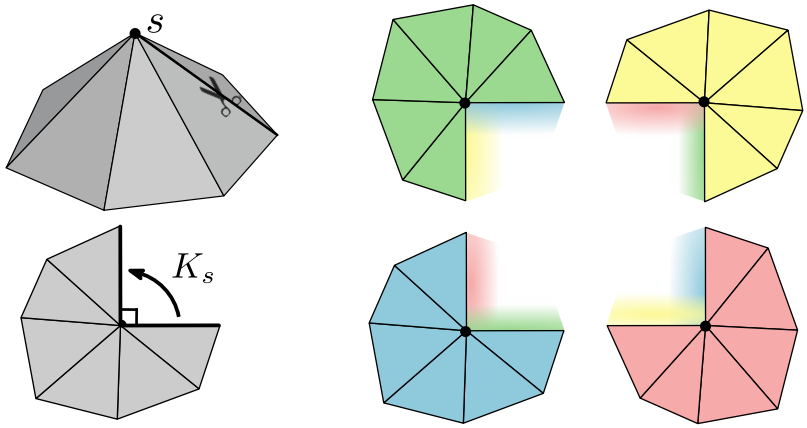


Figure 3.1: The triangle ring of a vertex s can always be isometrically flattened in the plane, creating an angle defect K_s (left). Creating a 4-covering of the ring consists in duplicating it four times and concatenating the versions together. When reassembled, this new ring is flat (i.e. has a defect multiple of 2π) if and only if K_s is a multiple of $\pi/2$.

element distortion or anisotropy. In other words, finding the best possible cone distribution for a given shape is still an open problem.

In this chapter, we investigate a first method for computing a cone distribution with its associated seamless parameterization. The goal is to be able to work with the cone positions and defects as well as the *uv*-coordinates in order to optimize some user-given quality criterion. As the cone distribution involves integer variables in the form of the indices $k \in \mathbb{Z}$ of the defects, we propose not to use a mixed-integer approach, but to optimize on a different space, namely the 4-covering of the input mesh, where the problem becomes completely continuous and can be optimized using quasi-Newton solvers like the L-BFGS [Liu and Nocedal 1989] algorithm.

In the context of quadmeshing, 4-coverings were first introduced by Kälberer et al. [2007]. The idea behind the transformation is simple: singular vertices having defect of $n\pi/2$ can have their triangle ring laid out in parameter space as a triangle fan with a matching angular opening (Figure 3.1, left). Then, by stitching together four versions of this fan (Figure 3.1, right), it is possible to build a new ring with a defect related to four times the original one. This means that as long as the original defect is a multiple of $\pi/2$, the defect of the new ring will be a multiple of 2π, which corresponds to a flat ring. In other words, all relevant singularity defect we want to consider all correspond to a single configuration in the 4-covering.

Based on this observation, we propose to compute a seamless parametrization by decomposing the input mesh as a set of overlapping 4-covering of rings and optimize a "flatness" energy function for each of them. This energy, based on the LSCM energy [Lévy et al. 2002], has a long history of applications in geometry processing, especially in the domain of conformal flattening and injective embedding of a surface in a 2D domain [Du et al. 2020; Xu et al. 2011]. Related work on these topics is covered by Sections 2.3.2, 2.3.3 and 2.4.4 of Chapter 2.

This chapter is organized as follows. In Section 3.1, we describe in more detail the 4-covering space we consider for our problem. In Section 3.2, we recall the LSCM algorithm and derive a specific flatness energy, function of the *uv*-coordinates as well as cotangents of the mesh. In Section 3.3, we describe how this flatness energy can be applied to the case of a 4-covering and describe a practical algorithm to optimize it. Finally, while our formulation of the cone distribution problem works mathematically, it failed to meet our expectations in practice due to numerical reasons we expose and discuss in Section 3.4.

3.1 4-covering of a Triangle Mesh

3.1.1 Decomposition as triangle rings

Let us begin by defining the space we will be working on for this whole chapter. Let $M = (V, E, T)$ be an input triangle mesh. Let $s \in V$ be a vertex and $\mathcal{R}(s) := (t_1, t_2, ...t_N)$ be the ring of triangles incident to s described in direct order around s.

To define the 4-covering of M, we first decompose M as disjoint ring sets $\mathcal{R}(s)$ by duplicating each triangle $t = (i, j, k)$ three times, once per incident vertex. Then, for each ring $\mathcal{R}(s)$, we create four identical versions of each triangle. We denote by $t_n^{s,c}$ the c-th copy ($c = 1, 2, 3, 4$) of triangle $t_n \in T$ in the ring of the considered vertex s. This construction and the notations are illustrated in Figure 3.2. The 4-covering of the ring of s (Figure 3.3a) is the triangle sequence $\mathcal{R}^{(4)}(s)$ defined as:

$$\mathcal{R}^{(4)}(s) := (t_1^{s,1}, ..., t_N^{s,1}, t_1^{s,2}, ..., t_N^{s,2}, t_1^{s,3}, ..., t_N^{s,3}, t_1^{s,4}, ..., t_N^{s,4}). \tag{3.1}$$

In other words, the four duplicated versions of the ring $\mathcal{R}(s)$ are concatenated in order to form the 4-covering $\mathcal{R}^{(4)}(s)$.

The 4-covering $M^{(4)}$ of the mesh M is then the union of all the 4-coverings of its triangle rings:

$$M^{(4)} := \bigcup_{s \in V} \mathcal{R}^{(4)}(s).$$

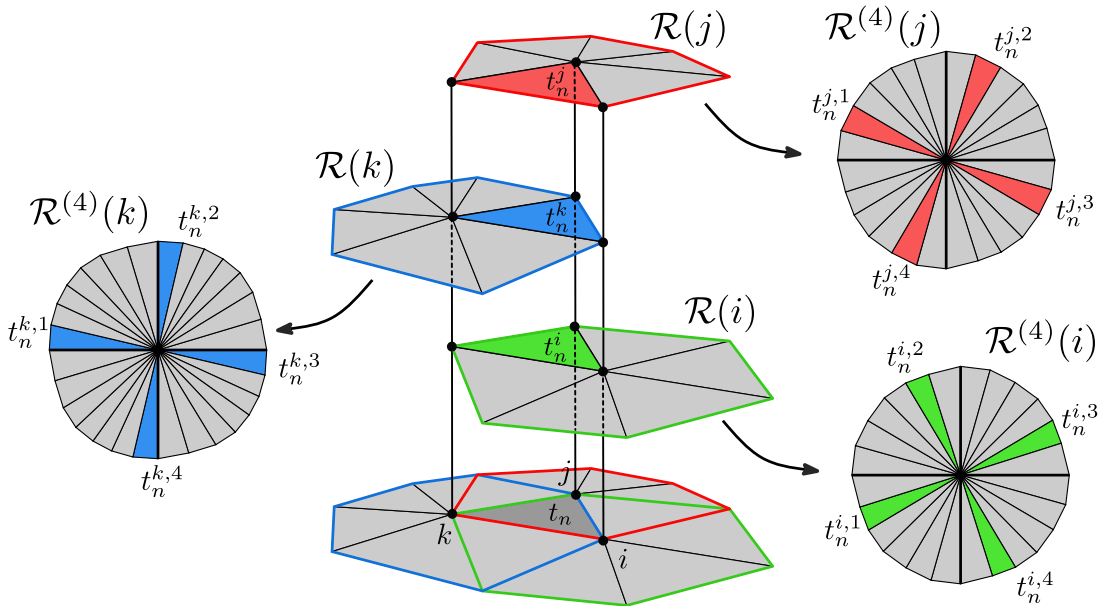
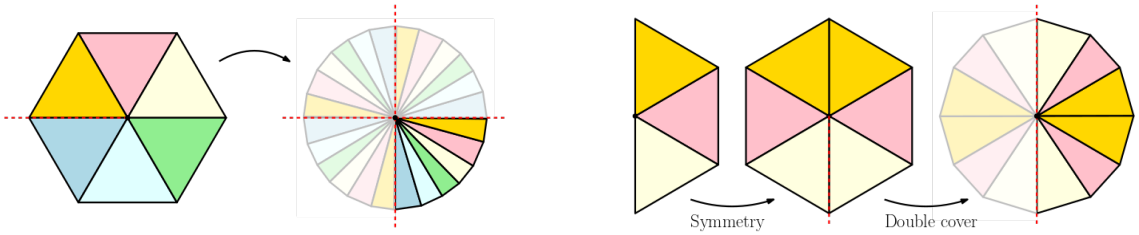


Figure 3.2: To build a 4-covering $M^{(4)}$ of a mesh, each original triangle is duplicated 12 times: 4 times for each of its 3 corners. Here, we represent this construction for a triangle $t_n = (i, j, k)$. For each of its corner $s \in \{i, j, k\}$, four copies of t_n noted $t_n^{s,1}, t_n^{s,2}, t_n^{s,3}$ and $t_n^{s,4}$ are present in the 4-covering $\mathcal{R}^{(4)}(s)$. Simple rings $\mathcal{R}(i), \mathcal{R}(j)$ and $\mathcal{R}(k)$ are represented geometrically here whereas 4-coverings $\mathcal{R}^{(4)}(i), \mathcal{R}^{(4)}(j)$ and $\mathcal{R}^{(4)}(k)$ are drawn in a combinatorial way.



(a) 4-covering of the ring of an interior vertex. (b) 4-covering of a ring on the boundary.

Figure 3.3: Combinatorial illustration of the 4-covering's construction.

Rings of boundary vertices For a boundary vertex $s \in V$, its initial ring in M was only a fan. The construction of its 4-covering ring is slightly different to account for the different definition of the angle defect in that case: we first symmetrize the fan to mimic a complete interior ring before performing a double-covering (Figure 3.3b):

$$\mathcal{R}^{(4)}(s) := (t_1^{s,1}, ..., t_N^{s,1}, \overline{t_N^{s,2}}, ..., \overline{t_1^{s,2}}, t_1^{s,3}, ..., t_N^{s,3}, \overline{t_N^{s,4}}, ..., \overline{t_1^{s,4}}). \tag{3.2}$$

This is equivalent to the 4-covering of interior vertices where triangles of even layers are symmetrized (as $\overline{t_n^{s,2}}$) and enumerated in reverse order.

Topological constraints Additionally, for a parametrization to be seamless on an arbitrary topology, one has to take into account defect accumulated along non-contractible cycles γ_c and boundary to boundary paths γ_b, represented as a loop and stripe of triangles respectively. Given a non-contractible cycle $\gamma_c = (t_1, ..., t_n)$, we define its 4-covering $\mathcal{R}^{(4)}(\gamma_c)$ similarly to an interior ring (Equation (3.1)). For a path γ_b linking two boundaries, we perform the same operation as for a boundary ring: first symmetrize the triangles and then apply a double-covering (Equation (3.2)).

The four kinds of rings considered in our space are depicted in Figure 3.4.

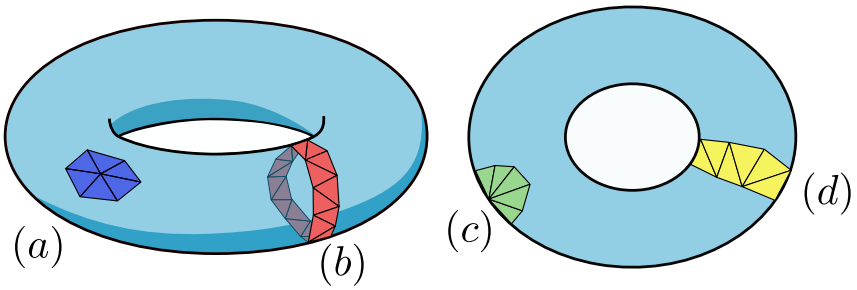


Figure 3.4: The four types of triangle rings considered in our approach. (a) interior vertex ring. (b) non-contractible cycle. (c) boundary vertex ring. (d) boundary to boundary path.

Overall, each initial triangle of M is duplicated at least twelve times in the 4-covering $M^{(4)}$: four times for each of its three corners, plus eventual versions if the triangle is part of a topological path. This duplication is necessary to ensure that the resulting space is a manifold mesh on which parametrization algorithms can be applied.

3.1.2 Properties of the 4-covering

Given a vertex $s \in V$, we denote by K_s its angle defect in M. We define an alternative notion of the angle defect on its 4-covering ring $\mathcal{R}^{(4)}(s)$, noted $K_s^{(4)}$, as:

$$K_s^{(4)} := 8\pi - \sum_{c=1}^{4} \sum_{n} \alpha_n^{s,c}$$

where $\alpha_n^{s,c}$ is the corner angle of triangle $t_n^{s,c}$ at vertex s. The construction of $M^{(4)}$ is justified by the following theorem:

Theorem 3.1. *If $K_s^{(4)} = 0$ mod 2π for all $s \in V$, and all 12 versions $t_n^{s,c}$ of triangle $t_n \in T$ have matching lengths in parameter space, then a seamless parametrization of M can be extracted from $M^{(4)}$.*

Proof. We give a proof of this in the case where the considered mesh is a topological sphere, that is to say when all rings are rings of an interior vertex.

Given a triangle $t = (i, j, k) \in T$, having its twelve versions have matching lengths means that we can define consistent coordinates $(u_i, v_i, u_j, v_j, u_k, v_k)$ in an arbitrary basis such that each of its versions $t^{s,c}$ has the same coordinates up to a translation and rotation. This means in particular that for a vertex s, the four triangles $t_n^{s,1}, t_n^{s,2}, t_n^{s,3}$ and $t_n^{s,4}$ have matching lengths and corner angles. The 4-covering ring $\mathcal{R}^{(4)}(s)$ can therefore be seen as four copies of the same ring $\mathcal{R}(s)$ assembled together. As a consequence, we can write the defect $K_s^{(4)}$ as:

$$K_s^{(4)} = 4\left(2\pi - \sum_{n} \alpha_n^{s,1}\right) = 4K_s.$$

It directly follows that if $K_s^{(4)} = 2n\pi$ for some integer $n \in \mathbb{Z}$, then $K_s = n\pi/2$. In other words, by considering the uv-coordinates defined as the common coordinates of triangle versions (which can be assembled along a tree), we have extracted a parametrization of the surface with quantized cones having defect multiple of $\pi/2$, that is to say a (rotationally) seamless parametrization. □

Given this setup in terms of combinatorics of the input mesh as well as the result of Theorem (3.1), our goal is to define an energy over the coordinates of triangles whose minimization will result in the defect of each 4-covering ring $\mathcal{R}^{(4)}(s)$ to be a multiple of 2π. Forcing each triangle versions to have matching lengths in the optimization of this energy will allow us to compute a seamless parametrization of our input surface.

3.2 Flattening Energy

In order to design a flattening energy function of the uv-coordinates, we take inspiration from the *least-square conformal maps* algorithm of Lévy et al. [2002]. This parametrization method, like many in the state of the art, is not able to differentiate a locally injective solution from a double or n-covering. This is considered a drawback since double coverings are undesirable in general. Yet, in our case, we will take advantage of this property to directly flatten our 4-coverings per ring and recover a cone distribution.

3.2.1 Least-Square Conformal Maps

Let us first dive into the theory and applications of conformal maps to define our flatness energy. Conformal maps are a class of functions that locally preserve angles and have been extensively used for the parametrization of surfaces with disk topology (see Section 2.3.2). Formally, let $\mathcal{M}$ be a smooth 2D manifold with disk topology and $\phi : \mathcal{M} \to \mathbb{R}^2$ be a differentiable function. ϕ is said to be conformal if it satisfies the *Cauchy-Riemann* equation:

$$d\phi_p(Jx) = Jd\phi_p(x) \tag{3.3}$$

for any point $p \in \mathcal{M}$ and tangent vector $x \in T_p\mathcal{M}$, where $J = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ is the $\pi/2$ rotation of the plane (Figure 3.5). If we decompose ϕ as its coordinate functions $(u\ v)^T$ with $u, v : \mathcal{M} \to \mathbb{R}$, we can rewrite Equation (3.3) as:

$$J\nabla u = \nabla v \tag{3.4}$$

In other words, the gradients ∇u and ∇v form a direct orthogonal basis at every point of the complex plane.

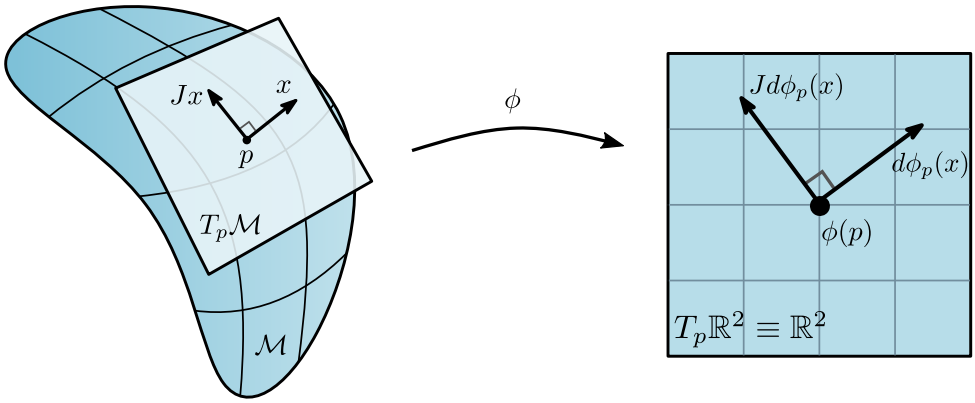


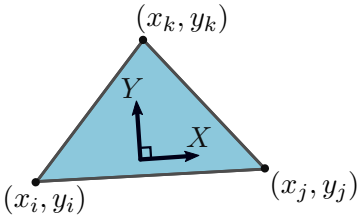
Figure 3.5: A conformal map is a differentiable function that locally preserves angles.

In essence, the LSCM algorithm [Lévy et al. 2002] proposes a direct discretization of Equation (3.4) on a triangle mesh $M = (V, E, T)$ with disk topology. More precisely, the parametrization ϕ is represented as the coordinates (u, v) in the plane of each vertex. Since we consider it to be piecewise linear per triangle (i.e., linearly interpolating corner coordinates inside triangles), the gradients $\nabla_t u$ and $\nabla_t v$ form two tangent vector fields that are constant per triangle t. Following Equation (3.4), the LSCM system simply requires that those gradients should form an orthogonal basis and optimizes the condition in the least-square sense to find:

$$\underset{u,v}{\operatorname{argmin}} \sum_{t \in T} E_t(u, v) \quad \text{where} \quad E_t(u, v) := A_t \left|\left|\nabla_t v - J\nabla_t u\right|\right|^2 \tag{3.5}$$

and A_t is the area of triangle t taken as a weight.

For a given triangle $t = (i, j, k)$ equipped with an arbitrary basis (X, Y) of its corresponding plane (inset Figure), the gradients $\nabla_t u$ and $\nabla_t v$ can be expressed as 2D vectors from xy-coordinates $p_i = (x_i, y_i)$ of the three vertices in the basis (X, Y) and uv-coordinates (u_i, v_i) [Botsch et al. 2010] as:



$$\nabla_t u = \frac{1}{2A_t} G_t \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix} \quad \text{where } G_t = \begin{pmatrix} y_j - y_k & y_k - y_i & y_i - y_j \\ x_k - x_j & x_i - x_k & x_j - x_i \end{pmatrix}$$

Therefore, we can rewrite the LSCM energy E_t per triangle as:

$$E_t(u, v) = A_t \left|\left| G_t \begin{pmatrix} v_i \\ v_j \\ v_k \end{pmatrix} - J G_t \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix} \right|\right|^2 \tag{3.6}$$

Using this formulation, Problem 3.5 can be solved like in the original work of Lévy et al. [2002] using a linear solver. However, the system is under-constrained as the uv-coordinates are defined up to a global scale and translation. One solution is to constrain a boundary edge to have fixed coordinates in uv-space. Another is to make use of an eigensolver to directly retrieve a unitary eigenvector of the system's matrix, as in [Mullen et al. 2008].

An important fact to remark is that the solution of the LSCM system is not conformal in general, as the residue of the least-square problem is non-zero. This is due to the fact that flattening a non-flat triangulated surface onto the plane is impossible without altering some of the angles. In fact, the following result is true:

Theorem 3.2. *The solution to the LSCM system has zero residue if and only if the angle defect K_s of each interior vertex s of the input mesh is equal to zero modulo 2π. In that case, the uv-coordinates solution of Equation* (3.5) *are an isometric parametrization up to a global scaling.*

Proof. If the LSCM energy is equal to zero, then for all triangles $t \in T$ of the mesh, we have $E_t = 0$.

For any edge $e = (i, j)$, the gradients ∇u and ∇v are continuous in the tangent direction of e by construction. The LSCM energy being zero implies that the gradients are also continuous in the direction normal to the edge and thus, completely continuous. Since the mapping ϕ from xyz-space to uv-space is a piecewise linear function, gradients are constant per triangle and continuity implies that they are constant everywhere. ϕ is therefore a global translation and scale. Since the parametrization has zero defect everywhere (except at boundaries), this is only possible if it was already the case for the mesh in xyz-space. □

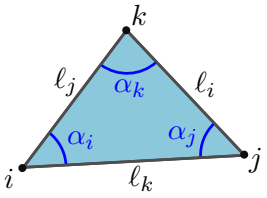
In other words, Theorem (3.2) states that the residual of the LSCM energy can be thought of as a numerical measure for "flatness" of a triangular surface. We now turn our attention of finding the closed form formulation of this energy.

3.2.2 Reformulation of the LSCM system

The first step for expressing the residual LSCM energy in closed form as function of the parametrization is to get rid of the dependency of E_t to a set of arbitrary orthonormal bases (X, Y) over all triangles. This amounts to reformulate the gradient operator G_t as a function of the uv-coordinates and the angles of the mesh. By adding this last dependency, we will be able to solve for uv-coordinates and corner angles and take advantage of Theorem (3.2) to compute a flat representation of each 4-covering rings. In short, we prove the following theorem:

Theorem 3.3. *For a triangle $t = (i, j, k)$, the LSCM energy per triangle E_t can be expressed as a function of the uv-coordinates and the corner angles α:*

$$E_t(u, v, \alpha) = \sum_{s \in \{i,j,k\}} \cot(\alpha_s)\ell_s(u, v) \; - \; A_t(u, v) \qquad (3.7)$$



*where $\ell_s(u, v)$ is the **squared** length of the side opposite to vertex s and $A_t(u, v)$ is the signed area of triangle t in parameter space. Moreover, when $E_t = 0$, the cotangents $\cot(\alpha)$ can be computed from the uv-coordinates.*

Proof. To reformulate Equation (3.6) into Equation (3.7), we first observe that for a triangle t, minimizing E_t is equivalent to solving $\nabla E_t = 0$ which is expressed as the system of linear equations:

$$\begin{cases} G_t^T G_t u = & G_t^T J G_t v \\ G_t^T G_t v = - & G_t^T J G_t u \end{cases} . \qquad (3.8)$$

Square matrices $G_t^T G_t$ and $G_t^T J G_t$ can be expressed using the xy-coordinates p_i, p_j, p_k of vertices i, j, k as:

$$G_t^T G_t = \frac{1}{4A_t^2} \begin{pmatrix} ||p_j - p_k||^2 & \langle p_j - p_k, p_i - p_k \rangle & \langle p_j - p_k, p_i - P_p \rangle \\ \langle p_j - p_k, p_i - p_k \rangle & ||p_i - p_k||^2 & \langle p_j - p_k, p_i - p_j \rangle \\ \langle p_j - p_k, p_i - p_j \rangle & \langle p_j - p_k, p_i - p_j \rangle & ||p_i - p_j||^2 \end{pmatrix} \qquad (3.9)$$

and

$$G_t^T J G_t = \frac{1}{2A_t} \begin{pmatrix} 0 & 1 & -1 \\ 1 & 0 & 1 \\ -1 & 1 & 0 \end{pmatrix} . \qquad (3.10)$$

Now, let $\alpha_i, \alpha_j, \alpha_k$ be the angles at vertices i, j, k in triangle t. We can express dot products and A_t in Equation (3.9) and (3.10) using trigonometric functions:

$$\begin{aligned} & \langle p_j - p_k, p_i - p_k \rangle & = \; & ||p_j - p_k||\,||p_i - p_k|| \cos(\alpha_k) \\ 2A_t = \; & (p_j - p_k) \times (p_i - p_k) & = \; & ||p_j - p_k||\,||p_i - p_k|| \sin(\alpha_k) \end{aligned}$$

which in the end leads to the following expression of $G_t^T G_t$:

$$G_t^T G_t = \frac{1}{2A_t} \begin{pmatrix} \cot(\alpha_j) + \cot(\alpha_k) & -\cot(\alpha_k) & -\cot(\alpha_j) \\ -\cot(\alpha_k) & \cot(\alpha_i) + \cot(\alpha_k) & -\cot(\alpha_k) \\ -\cot(\alpha_j) & -\cot(\alpha_i) & \cot(\alpha_i) + \cot(\alpha_j) \end{pmatrix} . \qquad (3.11)$$

Now, we can rewrite energy E_t using Equations (3.10) and (3.11):

$$\begin{aligned} E_t &= A_t \, ||G_t u - J G_t v||^2 \\ &= A_t (u^T G_t^T G_t u + v^T G_t^T G_t v - 2 u^T G_t^T J G_t v) \end{aligned}$$

$$E_t = \underbrace{\begin{pmatrix} \cot(\alpha_i) & \cot(\alpha_j) & \cot(\alpha_k) \end{pmatrix} . \begin{pmatrix} (u_j - u_k)^2 + (v_j - v_k)^2 \\ (u_i - u_k)^2 + (v_i - v_k)^2 \\ (u_i - u_j)^2 + (v_i - v_j)^2 \end{pmatrix}}_{D_t} - \underbrace{\frac{1}{2} \det \begin{pmatrix} u_j - u_i & u_k - u_i \\ v_j - v_i & v_k - v_i \end{pmatrix}}_{A_t}$$

The first term D_t of this expression is the dot product between the vector of cotangents and the vector of opposite squared lengths. If the cotangents correspond to the angles made by uv-coordinates, then this quantity is exactly the unsigned area of triangle t [Pinkall and Polthier 1996]. The other term is half the determinant of uv-coordinates of the triangle, that is to say the *signed* area A_t of triangle t. The energy we consider per triangle is simply the difference between the two. Written more compactly, this expression is equivalent to Equation (3.7).

Additionally, as $D_t \geqslant A_t$, the only possibility for the difference to be zero is for $D_t(u, v, \alpha)$ to effectively be the area of the triangle, meaning that cotangents $\cot(\alpha)$ should be the ones expressed as uv-coordinates:

$$\cot(\alpha_k) = \frac{(u_i - u_k)(u_j - u_k) + (v_i - v_k)(v_j - v_k)}{(u_i - u_k)(v_j - v_k) - (u_j - u_k)(v_i - v_k)}.$$

□

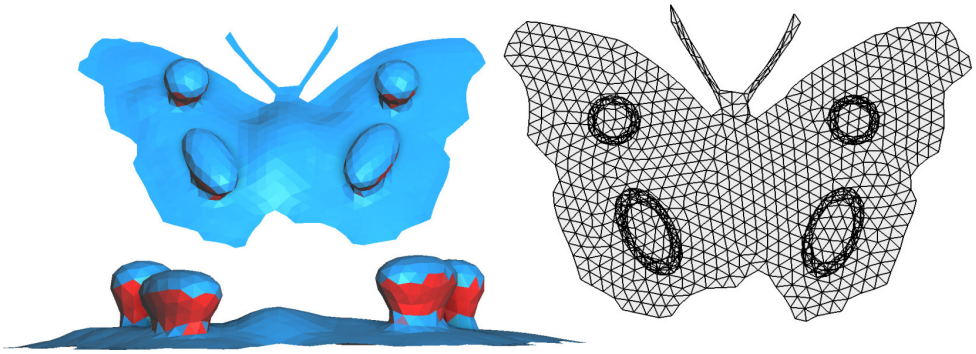
3.2.3 Energy Interpretation

According to Theorem 3.2 and the simple observation that $E_t \geqslant 0$, we can deduce that $E = \sum_t E_t$ is zero if an only if the considered mesh is flat with no flipped triangles. By fixing the boundary of a disk-topology mesh, minimizing E is equivalent to finding a foldover-free embedding of the mesh, that is to say a (locally) injective parametrization, as we demonstrate in Figure 3.6. This idea has been explored by Xu et al. [2011], where the authors proposed to minimize D_t by iteratively minimizing along the uv-coordinates (solving a linear system) and updating the cotangents. This method of resolution is, however, less suited to our case where we take into account the second term A_t. We instead consider the energy E as a (non-linear) function of both the uv-coordinates and the cotangents, and optimize simultaneously over this greater set of variables.

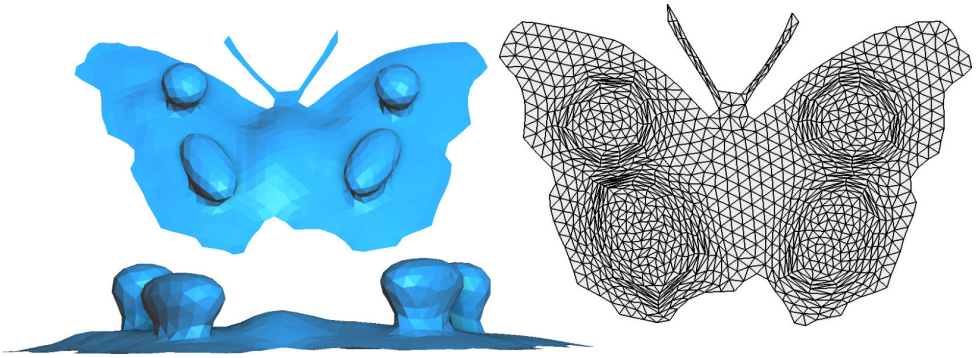
Fortunately, the fact that cotangents correspond to uv-coordinates at the minimum allows us to initialize them as the actual cotangents of the mesh but consider them as a set of variables that can be optimized freely. Provided convergence to the zero of the energy E_t, we are still assured to obtain the correct result.

3.3 Optimization of the Flattening Energy

Now that we have defined a flatness energy E_t per triangle (Equation (3.7)), we are interested in solving the following minimization problem:



(a) The initial parametrization is a projection onto the xy-plane and presents flipped triangles (in red)



(b) Minimizing E_T for all triangles lead to a parametrization with no flipped triangles

Figure 3.6: Given a fixed boundary, finding a zero of energy E is equivalent to finding a parametrization with no flipped triangles. (a) A 2D mesh with bumps is parametrized by projecting its vertices onto the xy-plane. (b) Minimizing E using a L-BFGS algorithm lead to a foldover-free parametrization with the same boundary.

$$\begin{aligned} \underset{u,v,\alpha}{\operatorname{argmin}} \quad & \sum_{t=(i,j,k)} \left(\sum_{s\in\{i,j,k\}} \cot(\alpha_s)\ell_s(u,v) \;-\; A_t(u,v) \right) \\ \text{subject to} \quad & \alpha_i^t + \alpha_j^t + \alpha_k^t = \pi \quad \text{for all } t \in T \end{aligned} \tag{3.12}$$

where we consider u, v and $\cot(\alpha)$ to be our sets of variables. The constraint on angles guarantees the validity of each triangle. Our goal is now to integrate this constraint in the optimization and find a closed form of the minimum over α in order to retrieve a formulation that depends only on the uv-coordinates.

3.3.1 Optimization Problem without 4-covering

In order to simplify the exposition, let us first derive our final energy expression on the mesh M and not $M^{(4)}$. The modifications needed to express the energy over a full 4-covering are postponed to Subsection 3.3.2. Meanwhile, we prove the following result:

Theorem 3.4. *Problem* (3.12) *is equivalent to:*

$$\underset{u,v}{\operatorname{argmin}} \sum_{t=(i,j,k)} \sqrt{2(\ell_i\ell_j + \ell_j\ell_k + \ell_k\ell_i) - \ell_i^2 - \ell_j^2 - \ell_k^2} \; - \; A_t(u,v). \tag{3.13}$$

Proof. The idea of the proof is to express the validity constraints $\alpha_i + \alpha_j + \alpha_k = \pi$ of angles in terms of cotangents that will then be integrated inside the energy using Lagrange multipliers.

Let us begin by proving a small lemma:

Lemma 3.5. *Given three numbers $a, b, c > 0$:*

$$a + b + c = \pi \quad \Leftrightarrow \quad \cot(a)\cot(b) + \cot(b)\cot(c) + \cot(c)\cot(a) = 1$$

This identity simply comes from the formula for the cotangent of a sum:

$$\cot(a+b) = \frac{\cot(a)\cot(b) - 1}{\cot(a) + \cot(b)}. \tag{3.14}$$

Writing $a + b = \pi - c$ and taking the cotangent of both sides leads to:

$$\frac{\cot(a)\cot(b) - 1}{\cot(a) + \cot(b)} = -\cot(c)$$

which gives us the correct identity up to some reordering. Given this expression in terms of cotangents, we can add this validity constraint to Problem 3.12. As it is not a linear constraint, we use the method of Lagrange multipliers, which consists in introducing a new variable λ and a new function $\mathcal{L}_t$ per triangle t to minimize over u, v, α and λ:

$$\mathcal{L}_t(u,v,\alpha,\lambda) := E_t(u,v,\alpha) + \lambda\left(\cot(\alpha_i)\cot(\alpha_j) + \cot(\alpha_j)\cot(\alpha_k) + \cot(\alpha_k)\cot(\alpha_i) - 1\right). \tag{3.15}$$

The method then calls for the cancellation of the derivative $\dfrac{\partial \mathcal{L}_t}{\partial \lambda}$ in order to get an expression of λ in terms of the other variables:

$$\lambda = \frac{1}{2}\sqrt{2(\ell_i\ell_j + \ell_j\ell_k + \ell_k\ell_i) - \ell_i^2 - \ell_j^2 - \ell_k^2} \tag{3.16}$$

λ can again be interpreted as the unsigned area of triangle t by Heron's formula. Now, as we are looking for the minimum over α, we can also cancel the partial derivatives $\dfrac{\partial \mathcal{L}_t}{\partial \alpha_i}$, $\dfrac{\partial \mathcal{L}_t}{\partial \alpha_j}$ and $\dfrac{\partial \mathcal{L}_t}{\partial \alpha_k}$. This leads to the classical expression of the cotangents in terms of lengths and triangle area [Eck et al. 1995], also referred to as the law of cosines:

$$\cot(\alpha_i) = \frac{\ell_j + \ell_k - \ell_i}{2\lambda}, \quad \cot(\alpha_j) = \frac{\ell_k + \ell_i - \ell_j}{2\lambda}, \quad \cot(\alpha_k) = \frac{\ell_i + \ell_j - \ell_k}{2\lambda} \tag{3.17}$$

Plugging these expressions into Problem (3.12) amounts at considering the minimum over α of the energy under the validity constraint. This gives us an equivalent minimization problem that does not depend on the cotangents anymore:

$$\underset{u,v}{\operatorname{argmin}} \sum_{t=(i,j,k)} \sqrt{2(\ell_i\ell_j + \ell_j\ell_k + \ell_k\ell_i) - \ell_i^2 - \ell_j^2 - \ell_k^2} \; - \; A_t(u,v)$$

which is exactly the expression of Problem (3.13). □

For ease of notations, we simplify the expression in Problem (3.13) by introducing the vector $\ell_t := (\ell_i \ \ell_j \ \ell_k)^T$ of **squared** lengths in triangle t and the matrix $H := \begin{pmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix}$. Then, we can compactly write our flattening problem as:

$$\underset{u,v}{\operatorname{argmin}} \sum_t \sqrt{\ell_t(u,v)^T H \ell_t(u,v)} \ - \ A_t(u,v). \tag{3.18}$$

3.3.2 Optimization Formulation over a 4-covering

Now that we reached our goal of finding an energy defined on individual triangles that measures the flatness of a mesh, let us move to the case of the 4-covering $M^{(4)}$. Following Theorem (3.1), we are interested in finding a solution of Problem (3.18) under the constraint that duplicated versions of the same triangle should have matching lengths.

Recall from Section 3.1 that in $M^{(4)}$, a triangle $t = (i, j, k)$ is duplicated into 12 versions $t^{s,c}$ for $s \in \{i, j, k\}$ and $c \in \{1, 2, 3, 4\}$. Going back to Expression (3.12) of the optimization problem in terms of u, v and cotangents, we would like to find the minimum of the sum over all possible versions of all possible triangles. In practice, this would require the definition of uv-coordinates per triangle version $(u^{s,c}, v^{s,c})$ for each vertex, as well as a different set of cotangents $\cot(\alpha^{s,c})$. However, given the fact that we want the lengths of all versions to match, we instead define a single set of cotangents shared by all versions of triangle t. This amounts to fixing all of its versions to have matching shapes up to some scale, translation and rotation, and therefore matching lengths at the minimum of the energy.

By following the same steps as in Section 3.3.1, we prove our final result:

Theorem 3.6. *Computing a seamless parametrization of a mesh M is equivalent to solving the following optimization problem on $M^{(4)}$:*

$$\underset{u,v}{\operatorname{argmin}} \sum_{t=(i,j,k)} \left(\sqrt{\sum_{\substack{s\in\{i,j,k\} \\ c\in\{1,2,3,4\}}} \ell^{s,c}(u,v)^T H \ell^{s,c}(u,v)} \ - \sum_{\substack{s\in\{i,j,k\} \\ c\in\{1,2,3,4\}}} A_{t^{s,c}}(u,v) \right) \tag{3.19}$$

*where $\ell^{s,c} = \left(\ell_i^{s,c} \ \ell_j^{s,c} \ \ell_k^{s,c}\right)^T$ is the vector of **squared** edge lengths in version $t^{s,c}$ of triangle $t = (i, j, k)$. and $A_{t^{s,c}}$ is its signed area (both depending on u and v).*

Proof. Notice that Problem (3.19) is simply Problem (3.18) where we substituted lengths and areas by sums over all versions of a triangle. To arrive at this result, let us start by expressing the optimization over u, v and α using our notations for the 4-covering to get a modified version of Problem (3.12). For the sake of simplicity of notations, let us consider the case of a single triangle $t = (i, j, k)$ and its 12 versions:

$$\underset{u,v,\alpha}{\operatorname{argmin}} \sum_{\substack{s\in\{i,j,k\} \\ c\in\{1,2,3,4\}}} \sum_{m\in\{i,j,k\}} \cot(\alpha_m) \ell_m^{s,c}(u,v)^2 \ - \ A_{t^{s,c}}(u,v) \tag{3.20}$$

Since the cotangents are common to all triangle versions, they can be factored out of the sum. The problem then becomes:

$$\underset{u,v,\alpha}{\operatorname{argmin}} \sum_{m\in\{i,j,k\}} \left[\cot(\alpha_m) \cdot \sum_{\substack{s\in\{i,j,k\}\\ c\in\{1,2,3,4\}}} \ell_m^{s,c}(u,v)^2 \right] - \sum_{\substack{s\in\{i,j,k\}\\ c\in\{1,2,3,4\}}} A_{t^{s,c}}(u,v) \tag{3.21}$$

From Problem (3.21), the same arguments present in the proof of Theorem (3.4) can be applied. The Lagrange multiplier works out similarly except the sum of square lengths over all versions of the triangle replaces single square lengths. Overall, we obtain:

$$\underset{u,v}{\operatorname{argmin}} \sqrt{\sum_{\substack{s\in\{i,j,k\}\\ c\in\{1,2,3,4\}}} \ell^{s,c}(u,v)^T H \ell^{s,c}(u,v)} - \sum_{\substack{s\in\{i,j,k\}\\ c\in\{1,2,3,4\}}} A_{t^{s,c}}(u,v)$$

which is the single triangle version of Problem (3.19), which can be immediately summed over all triangles of M.

Now, since the optimization of Problem (3.19) will result in a flat 4-covering, we know from Theorem (3.1) that it will result in a seamless parametrization of the initial mesh M. □

3.3.3 Practical Algorithm

As all global minima of Problem (3.19) are seamless parametrizations of M, we now focus on its minimization in practice. Our set of variables consists of 12 triplets of uv-coordinates per triangle t of the input mesh M, each corresponding to one version of the triangle. As the final minimization problem does not involve cotangents explicitly anymore, we do not need them as variables.

Initialization The objective function in Problem (3.19) is obviously non-convex thus the final result is dependent on the initialization of the variables. We propose to initialize each 4-covering ring $\mathcal{R}^{(4)}(s)$ by solving the LSCM system (Equation (3.5)) and recovering uv-coordinates for the four involved versions of each triangle. As the objective function is only expressed in terms of lengths and areas in parameter space, it is invariant by translation and rotation of the uv-coordinates. We are therefore able to set the uv-coordinates corresponding to vertex s in $\mathcal{R}^{(4)}(s)$ at zero by convention and remove some variables.

Regularization For minimization, we require to compute gradients of our energy function. Since the square root in the first term has infinite derivative at zero, we regularize the expression by instead using the function $x \mapsto (x^2+\varepsilon)^{\frac{1}{4}}$ with $\varepsilon = 10^{-10}$.

Reconstruction Once the energy reaches zero, there are several possibilities to recover a cone distribution and an associated seamless parametrization. A first approach consists in reconstructing each vertex ring by extracting the first layer of its 4-covering and then stitching them together along a spanning tree. This, however, can be numerically unstable as any error made on a ring is propagated along the tree.

Another approach is to compute the final cotangents of all angles in the mesh from the uv-coordinates using the formulas of Equation (3.17). This allows us to compute the angle defect at each vertex and recover a valid cone distribution via rounding. Then, as for any cone parametrization (see Section 2.3.6), the mesh can be cut along a spanning tree as well as along non-contractible cycles to recover a disk topology. We compute the cotan Laplacian associated

with our final cotangents (following the definition of Equation (2.5) in Chapter 2) and solve for the eigenvector associated to the smallest eigenvalues, following the method of Mullen et al. [2008]. This more complicated yet more stable approach is the one used to produce the results of Figure 3.7.

3.4 Results and Discussion

We implemented the minimization of Problem (3.19) in python 3. For the optimizer, we used a limited memory BFGS algorithm [Liu and Nocedal 1989] as provided in the *scipy* library.

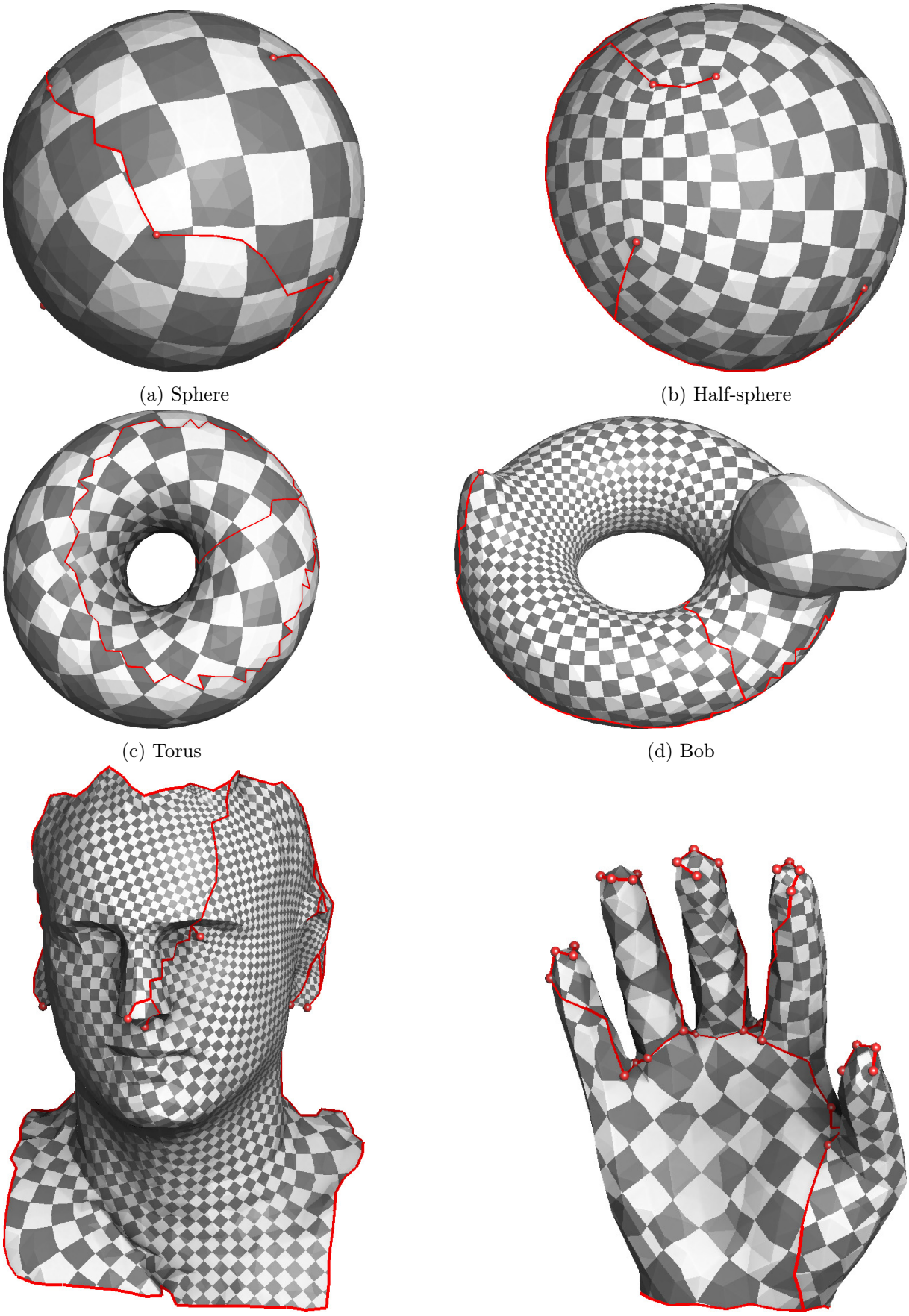
In Figure 3.7, we show parametrizations and cone distributions obtained by our algorithm on some surface triangular meshes. Although we are able to converge to correct results in some simple cases, this method of producing global parametrization proved to be numerically impractical for several reasons. Firstly, while we proved that zeros of our energy correspond to correct cone distributions, the non-linear nature of the optimization provide no guarantee of convergence towards said zeros. In practice, we observe that, when initialized with zero defect, rings hardly converge to a different defect. This is due to the profile of the energy in function of the defect, as plotted in Figure 3.8, that presents local maxima at $k\pi/2 + \pi/4$ which are hard to overcome without the ring area going through a zero. In practice, the optimization gets easily stuck in saddle points where all rings share a small amount of defect.

In addition to this problem, the method provides little to no control on where the singularity cones will appear. Mesh quality indeed play a great role in this final distribution, as rings associated with vertices of small valence present a smaller energy when their defect is close to $k\pi/2 + \pi/4$. thus encouraging singularity cones to appear at those vertices. This is particularly visible on the sphere model in Figure 3.7a, where singularity cones appear on the eight vertices of valence 5.

3.5 Conclusion

In this chapter, we investigated a combinatorial transformation over a triangle mesh to compute a quantized cone distribution and a seamless parametrization. Given an artificially built 4-covering of each ring, we derived an energy function whose zeros correspond exactly to zero angle defect, which in the original mesh corresponds by construction to defects multiple of $\pi/2$.Thanks to these two elements, we were able to formulate the problem of cone distribution as a continuous non-linear optimization. But while the minimization of our energy was theoretically equivalent to finding a valid parametrization, the numerical optimization turned out to be impractical with difficulties in convergence as well as a lack of control on the final result. As this method was based on a combinatorial trick, its dependence to the underlying geometry of the mesh comes with no surprise, which is an undesirable feat of such an algorithm.

In the next chapter, we will turn our attention to another formulation of the problem of seamless parametrization and cone placement that avoid the above issues and allows for a better control on the final parametrization's properties.



(a) Sphere

(b) Half-sphere

(c) Torus

(d) Bob

(e) PhD advisor

(f) Hand

Figure 3.7: Parametrizations obtained via our algorithm on various models. Singularity cones are depicted as red sphere.

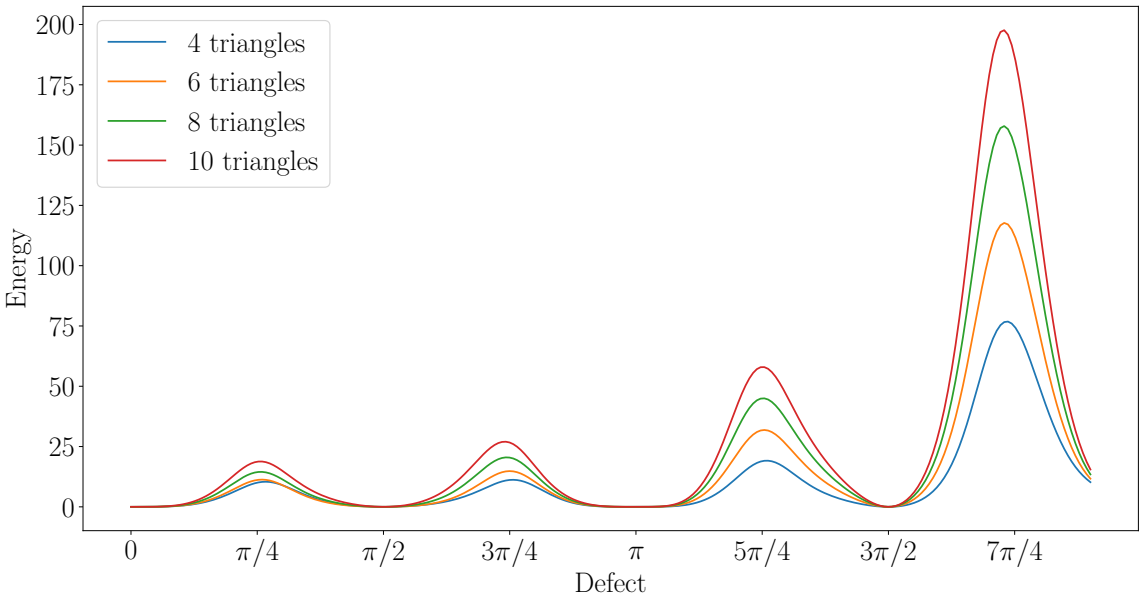


Figure 3.8: Plot of the energy in function of the angle defect K_s for a vertex ring with different numbers of triangles. As expected, zeros of the energy are found at multiples of $\pi/2$. However, the value of local maxima at $k\pi/2 + \pi/4$ depend on the valence n of the vertex.

4

The Method of Moving Frames for Surface Global Parametrization

Contents

4.1 Related Work . . . 88
4.2 Smooth Formulation . . . 89
4.2.1 Cartan's Method of Moving Frames . . . 89
4.2.2 Cartan's Method of Symmetric Moving Frames . . . 91
4.2.3 Parametrization Design . . . 92
4.2.4 Overview . . . 93
4.3 Discrete Setting . . . 93
4.3.1 The Cayley Map . . . 94
4.4 Local Frames . . . 94
4.4.1 Vertex-Based Frame Fields and Singular Triangles . . . 94
4.4.2 Parallel Transport From Levi-Civita connection . . . 95
4.4.3 Discrete Frame Field Condition . . . 95
4.4.4 Quantized Singularity Cones Fields . . . 96
4.5 First Structure Equation . . . 96
4.5.1 Vertex Charts . . . 96
4.5.2 Discrete Structure Equation . . . 97
4.5.3 Local Injectivity . . . 98
4.6 Quantized Cone Parametrization . . . 98
4.6.1 Parametrization Reconstruction . . . 98
4.6.2 Theoretical Guarantees . . . 100
4.7 Boundary and Feature Edge Adaptation . . . 102
4.8 Numerical Optimization . . . 103
4.8.1 Non-Linear Least-Squares . . . 104
4.8.2 Distortion Energy . . . 105
4.8.3 Initialization . . . 106
4.8.4 Parameter Choice . . . 108
4.9 Applications and Evaluation . . . 109
4.9.1 Seamless Parametrization . . . 109

4.9.2 Other Applications 110
4.9.3 Comparison With Other Methods 112
4.10 Conclusion and Perspectives 114

Computing a global seamless parametrization is a challenging problem in geometry processing, mostly due to the fact that any attempt at solving it needs to determine two very different sets of variables. One the one hand, a seamless parametrization is defined by a discrete set of singular points (or cones) that concentrate all the curvature in multiples of $\pi/2$. On the other hand, the actual mapping needs to be optimized to account for specific constraints depending on the application, such as alignment with feature edges or creases, local sizing of elements or controlled distortion. Yet, the mapping's topology is determined by the cone distribution, which can drastically alter its quality. In particular, finding cone positions minimizing the mapping's distortion is challenging and often leads to sophisticated optimization problems.

To overcome this apparent complexity, there exists two broad classes of algorithms. The first relies on two steps: first choosing the singularity locations using a cross field or other proxies and then compute a *global rotationally seamless parametrization*. This procedure, however, loosens the link between cone placement and the distortion of the final result. As a consequence, a quadmesh extracted from this parametrization can present quads that stray far away from squares because its connectivity was optimized almost independently of its geometry. The second type of methods restricts the parametrization to simpler deformations (typically conformal) allowing for simple distortion estimation but ends up using either greedy algorithms or highly specialized integer optimization algorithm for cone placement.

This is where differential geometry tools come to the rescue. *Cartan's method of moving frames* provides a rich theory to design and describe local deformations. This framework uses local frames as references, making local coordinates translation and rotation invariant [Lipman et al. 2007]. Furthermore, the *first structure equation* provides a necessary and sufficient condition to the existence of an embedded surface. This equation describes how differential coordinates should change relative to the frame's motion, effectively removing all influence of the ambient coordinate systems on the deformation.

In this chapter, we extend Cartan's method to *singular* frame fields and we prove that any solution of the derived structure equations is a valid cone parametrization. Most importantly, we provide a vertex-based discretization of the smooth theory which provably preserves all its properties. The absence of a global coordinate system allows us to compute parametrizations without prior knowledge of the cut positions and to automatically place quantized cones optimizing for a given distortion energy. Moreover, we study useful user prescribed constraints, such as feature or boundary alignment and cone locations as shown in Figure 4.1.

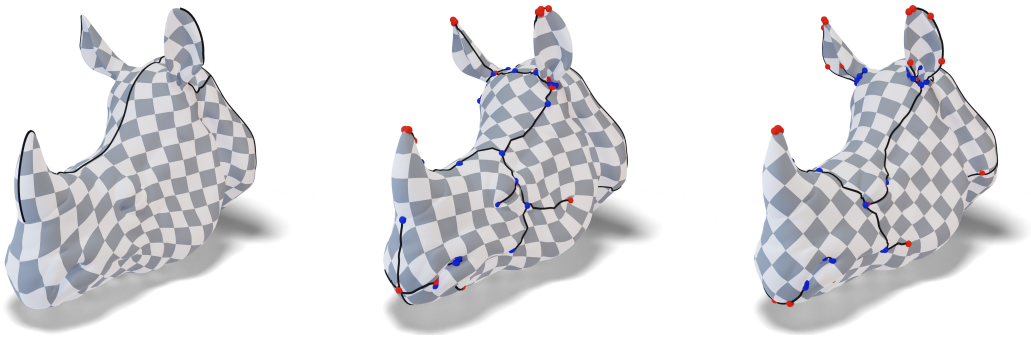
Our technical contributions include:

1. A formalization of surface parametrization using Cartan's method of moving frames;

2. An extension of Cartan's method to cone parametrization;

3. A versatile tool allowing all types of constraints as-well-as simple free-boundary parametrization;

4. An algorithm for simultaneously computing a parametrization and quantized cone positions without integer variables.



No cones
Scale = 1.041
Stretch = 1.413

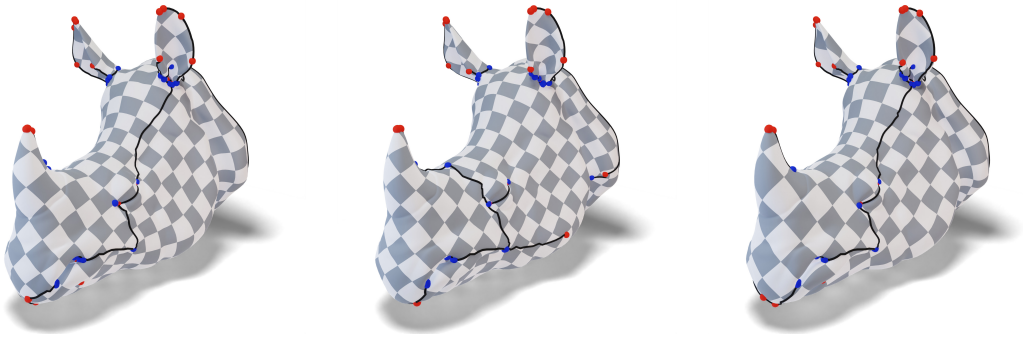
(a) Free Boundary

146 cones
Scale = 1.039
Stretch = 1.420

(b) Curvature

80 cones
Scale = 1.015
Stretch = 1.200

(c) Smooth Frame Field



80 cones
Scale = 1.015
Stretch = 1.188

(d) min ARAP

86 cones
Scale = **1.000**
Stretch = 1.246

(e) min AREA

78 cones
Scale = 1.083
Stretch = **1.079**

(f) min LSCM

Figure 4.1: Our discretization of the moving frames method allows us to *simultaneously* place singularity cones and compute a global parametrization in a single optimization problem. This figure presents our parametrizations of the *Rhino head* model computed for a variety of constraints: (a) by preventing cones from appearing, thus leading to a free boundary parametrization with user-defined cuts, (b) by following the topology of the principal directions of curvature cross field, (c) by following the topology of a smooth frame field [Viertel and Osting 2018], (d) by penalizing non-isometric deformations, (e) by minimizing area distortion, (f) by penalizing non-conformal deformations, effectively reducing stretch distortion. We represent positive $\pi/2$ cones in red and negative $-\pi/2$ cones in blue. Distortion measurements are computed as a mean over all triangles weighted by area.

4.1 Related Work

Cartan and Geometry Processing

In geometry processing, Cartan's method of moving frames has drawn a lot of attention for its rotation and translation invariant representation of metric and curvature [Lipman et al. 2005]. In a follow-up work, Lipman et al. [2007] use this representation to design volume preserving and as-isometric-as possible deformations. However, these works do not consider the structure equations. Wang et al. [2012] also design general deformations and provide discrete conditions, akin to the structure equations, for the existence of a deformation. However, in practice, frames are computed independently of the deformation and structure equations are never used. As a consequence, these methods are limited to suboptimal solutions and if they consider parametrization as an application, cones are out of scope.

Previous work using moving frame theory mostly focuses on deformations of volumes or surfaces with normals. In this context, the second structure equation is non-linear and describing a discrete equivalent of it is in itself a challenge [Corman and Crane 2019]. In contrast, we only study intrinsic properties of surfaces where the second structure equation is a simple linear equation [Crane et al. 2010]. The additional information provided by the *first* structure equation, which relates the local frames to the local deformation, allows us to compute maps from a surface to the plane.

In this work, the discretization of the structure equations is achieved by decomposing the mesh into disjoint vertex-based charts. We show that a valid parametrization can be constructed from any solutions of these discrete equations. No such guaranty can be given when using the vertex-based discretization of differential operators proposed by Liu et al. [2020].

Cross Field Based Parametrizations

Cross Fields are an extremely useful tool for seamless parametrization. We refer to Section 2.4.3 of Chapter 2 for a general overview of the topic and related works. In this family of methods, the *Simplex Assembly* algorithm of Fu and Liu [2016] in particular takes an ingenious approach: they divide the input mesh into a set of independent triangles and constrain adjacent simplices to be equal up to the cross field rotation. This simple constraint allows them to compute seamless global parametrizations for triangle-based cross fields*without* prior knowledge of the parametrization cuts. In this work, we use a similar formulation but adapted to cross field defined at vertices so that singularities are not bound to appear at vertices.

While convenient, cross field based approaches have one major drawback: not all cross fields can generate a valid parametrization. In particular, some can exhibit limit cycles creating degenerated triangles in parameter space. Many solutions have been proposed to circumvent this issue. Limit cycles can be detected and fixed locally by adding a singularity dipole [Myles et al. 2014]. However, this solution stays local and does not take into account the global problem of minimizing the distortion.

Diamanti et al. [2015] provide a system of necessary and sufficient equations for the existence of a seamless cone parametrization. The Jacobian of the deformation is expressed in a symmetry invariant way using the PolyVector representation. Like ours, this representation allow them to compute a cone parametrization freely placing 1/4-index singularity minimizing a distortion criterion. However, in practice, the algorithm is prone to add multiple singularities, even when starting from a cross field with integrable singularities.

Works on Chebyshev Nets have also introduced algorithms able to place singularities at the same time as computing geometry for the simple reason that these nets do not always exist on

general surfaces and often require specific cone placement. Sageman-Furnas et al. [2019] use a PolyVector representation but its integrability condition depends on the frame field matching, thus requires an integer optimization. Moreover, a post-processing step is needed to compute the parametrization as the construction is not exact. Later works, such as [Liu et al. 2020], alternate between computing a parametrization and a frame field are able to slowly move the singular points.

Maybe closer to our representation, Pluta et al. [2021] introduce the so-called "coordinate power fields". This representation simply relies on the definition of seamless parametrization: edges of adjacent triangles must be equal up to $\pi/2$ rotations. To avoid integer variables they raise edge vectors in the complex plane to the fourth power. In this setup, singularities are unlikely to appear as it would require creating a new cut along which all the variables must change. In our setting, singularities can be created easily by adjusting the frame field local rotations.

Cone Metric Deformation

Overall, a cross field is a convenient way of placing *quantized* singularities necessary for quadmeshing, however, it is not the only way to solve this problem. Most alternatives rely on conformal deformations as they offer a simple relationship between area distortion and cone positions. We refer to Sections 2.3.6 and 2.4.4 of Chapter 2 for an overview.

All these methods suffer from the same problem as cross fields: not all sets of cones admit a valid parametrization. For surfaces without boundary, prescribed cones satisfying the Gauss-Bonnet theorem are always feasible [Campen et al. 2019; Levi 2021] and a few obstructions exist for controlling non-contractible cycles [Shen et al. 2022]. However, the existence of boundaries or feature curves can lead to situations where quantized cones and Gauss-Bonnet theorem are no longer sufficient conditions. It appears to us that the only way to be certain that a set of singularities is consistent with a seamless mapping is to *compute* the underlying parametrization.

4.2 Smooth Formulation

Let us start by describing the theoretical background used to build our global parametrization technique, which is, at its core, a discretization of Cartan's method of moving frames. This simple and intuitive framework only uses two ingredients: local frames and Jacobian matrices expressed in local coordinates. The theory provides two *structure equations* precisely describing how these two quantities must relate in order to create a valid parametrization of the surface.

In all the chapter, Euclidean planes, either as parametric spaces or tangents to a surface, will be identified to the space of complex numbers $\mathbb{C}$. Orthogonal frames will be represented as elements of the space $\mathbb{U}$ of *unit* complex number.

4.2.1 Cartan's Method of Moving Frames

Although Cartan's theory is a very broad topic [Sharpe 1997], we will limit ourselves to the study of a mapping $\phi : \mathcal{M} \to \mathbb{C}$ sending a compact smooth orientable surface $\mathcal{M}$ to the complex plane. As a first step, we will further ask that $\mathcal{M}$ has disk topology. The results of this section can be found in the translation of Cartan's lecture on Riemannian geometry [Cartan et al. 2001].

Cartan's key idea is to assign, at each point of a surface, an orthogonal frame represented by a unit complex number $z : \mathcal{M} \to \mathbb{U}$ that will be used as a local reference system. The differential map $\mathrm{d}\phi : T\mathcal{M} \to T\mathbb{C} \equiv \mathbb{C}$, mapping tangent vectors of $\mathcal{M}$ to vectors in parameter space, is itself

projected into the local frame leading to the complex-valued 1-form $\sigma : T\mathcal{M} \to \mathbb{C}$ as illustrated in Figure 4.2. Cartan decomposes the differential map as a product of frame and local deformation:

$$(\mathrm{d}\phi)_p = z_p \sigma_p. \tag{4.1}$$

The complex-valued 1-form σ_p is nothing more than a representation of the Jacobian of the deformation at point $p \in \mathcal{M}$ in the coordinate system z_p.

Obviously, the frame field z and the local deformation tensor σ cannot be chosen arbitrarily and must depend on the frame's rotation. The main results of Cartan [Cartan et al. 2001, Thm. 1. p. 38] is that there exists a mapping ϕ and a frame field z solution of the system Equations (4.1) if and only if a system of two differential equations is satisfied.

The only requirement on the reference field z is that it should be smooth and unit norm. Thus, locally there exists a smooth angle function θ such that $z = \exp(\imath\theta)$. The frame's speed of rotation in direction X is given by differential form $\omega(X) := \mathrm{d}\theta(X)$. Therefore, the frame evolution is entirely captured by the differential equation:

$$\mathrm{d}z = \imath\omega z. \tag{4.2}$$

Equation (4.2) is well-known in geometry processing for computing cross fields [Crane et al. 2010; Ray et al. 2008] or even quad or stripe patterns [Knöppel et al. 2015; Ray et al. 2006]. As proved by Cartan et al. [2001]; Sharpe [1997], the frame field is uniquely determined by its rotation speed ω.

The evolution of deformation form σ is by definition deeply related to the evolution of the frame on the surface as it must compensate for the frame's rotation. The *first structure equation* describes exactly this relationship and, as proved in Thm. 4.1, it is also a necessary an sufficient condition for the existence of the mapping f defined in Equation (4.1).

Theorem 4.1. *Given a frame field whose rotation speed is ω, there exists a unique solution of the system* (4.1) *up to translation/rotation if and only if σ is solution of Cartan's* first structure equation*:*

$$\mathrm{d}\sigma + \imath\omega \wedge \sigma = 0. \tag{4.3}$$

Proof. The structure equation is equivalent to showing that $z\sigma$ is a closed-form:

$$\mathrm{d}(z\sigma) = \mathrm{d}z \wedge \sigma + z\mathrm{d}\sigma = z(\imath\omega \wedge \sigma + \mathrm{d}\sigma).$$

Since $\mathcal{M}$ is assumed to be disk-like, $z\sigma$ is exact if and only if it is closed.

A proof in a more general context can be found in [Cartan et al. 2001, Thm. 1, p. 38]. □

If the differential map and the deformation tensor σ differ by a rotation, they still share many properties. For instance, the local change of area is invariant by rotation and can be computed directly from σ. For parametrizations, we would like to obtain locally injective maps, or stated differently that the local areas stay positive. One way to ensure this is by checking that σ does not change the sign of the cross product between tangent vectors:

$$\star\mathrm{Im}(\bar{\sigma} \wedge \sigma) > 0. \tag{4.4}$$

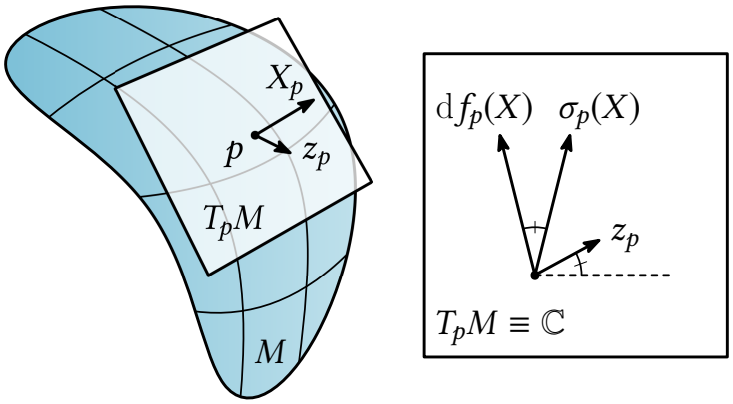


Figure 4.2: Cartan's method of moving frames projects the differential map $\mathrm{d}\phi$ at a point $p \in \mathcal{M}$ in a local basis $(z_p, \imath z_p)$ to obtain a deformation σ_p in the local referential.

4.2.2 Cartan's Method of Symmetric Moving Frames

So far, Cartan's method provided us with two integrability conditions for disk-like domains. In this section, we will extend this method to *global* parametrizations by allowing cone metrics — metrics with zero Gaussian curvature everywhere except at a few singular points. We will show that simply allowing symmetric frame fields is enough to guarantee parametrization with cone angles matching the singularity indices.

One obstruction to extending the theory to general topology is that, in general, we cannot define a regular reference field everywhere. Thus, we will consider frame with rotational symmetry [Palacios and Zhang 2007]. Two frames are deemed equivalent if they are equal up to a rotation in the symmetry group Γ. As shown in Crane et al. [2010], these symmetric fields still define a rotation speed ω and Equation (4.1) still admits a solution but this time within the quotient space $\mathbb{U}/\Gamma$.

Interestingly, the first structure equation Equation (4.3), obtained by differentiating Equation (4.1), does not change as it only depends of the frame's rotation speed.

Now we can define a global parametrization of a "cut" manifold $\mathcal{M}^c$. The surface $\mathcal{M}^c$ is defined by puncturing $\mathcal{M}$ at singular points and "cutting" the domain in order to recover a disk topology. We are able to show, in Thm. 4.2, that any solution of the first structure equation is a global parametrization of $\mathcal{M}^c$.

Theorem 4.2. *Given a 1-from σ satisfying Equation* (4.3) *and a symmetric frame field solution of Equation* (4.2) *in the quotient space $\mathbb{U}/\Gamma$, there exists a mapping $\phi : \mathcal{M}^c \to \mathbb{C}$ such that:*

$$\mathrm{d}\phi_p = z_p \sigma_p, \quad \forall p \in \mathcal{M}^c.$$

Moreover, the image of a pair of vectors tangent to the cuts are equal up to a rotation of the symmetry group Γ in parameter space.

Proof. The proof will follow three steps: 1) we demonstrate that there is a mapping ϕ satisfying Equation (4.1) in $\mathcal{M}^c$, 2) we show that σ is continuous away from singular points even though the frame field is discontinuous in $\mathbb{U}$ and 3) the vector tangent to the cuts are equal up to a rotation due to the discontinuity in the frame z.

First, we show the existence of a mapping ϕ solution of Equation (4.1). By construction, there exists a *smooth* frame field $z : \mathcal{M}^c \to \mathbb{U}$ whose Darboux derivative is ω. Since Equation (4.3)

holds true, the form $z\sigma$ is closed. Using the Poincaré lemma, we conclude that $z\sigma$ is exact on $\mathcal{M}^c$ as it is simply connected. Therefore, there exists a map $\phi : \mathcal{M}^c \to \mathbb{C}$ solution of:

$$\mathrm{d}\phi_p = z_p\sigma_p, \quad \forall p \in \mathcal{M}^c.$$

Secondly, we are going to prove that, even though z is *discontinuous* in $\mathbb{U}$, there always exists a form σ satisfying the first structure equation Equation (4.3) which is *smooth* away from singularities and independent of z and ω. To do so, we restrict ourselves to conformal deformations by choosing $\sigma = e^u \mathrm{d}x$ where $\mathrm{d}x = \mathrm{d}x_1 + \imath \mathrm{d}x_2$ is a local 1-form orthonormal basis. Injecting this into Equation (4.3), we show that the log-scale factor u is solution of the system of equations:

$$\begin{aligned} \mathrm{d}u \wedge \mathrm{d}x_1 + \omega \wedge \mathrm{d}x_2 = 0, \\ \mathrm{d}u \wedge \mathrm{d}x_2 - \omega \wedge \mathrm{d}x_1 = 0. \end{aligned}$$

The only solution of this system is: $\omega = - \star \mathrm{d}u$. The frame field rotation ω uniquely defines the frames up to global rotation [Sharpe 1997] and can be related to the Gaussian curvature K and the frame cones points Ω by the equation [Corman and Crane 2019; Crane et al. 2010]:

$$\mathrm{d}\omega = -K + \Omega.$$

Therefore, u is solution of the Poisson's equation:

$$\Delta u = K - \Omega. \tag{4.5}$$

Equation (4.5) always has a smooth solution away from singular points. Thus, σ is locally smooth away from singularities and is independent of the frame field discontinuities.

Third, we are going to study the properties of the parametrization cuts. Let $p \in \mathcal{M}$ be a point at a cut and p_+, p_- are the corresponding points on each side of the cut in $\mathcal{M}^c$. By construction, there exists a frame field $z : \mathcal{M}^c \to \mathbb{U}$ whose rotation speed ω is smooth on $\mathcal{M}^c$ but discontinuous on $\mathcal{M}$. At a discontinuity, there exists a transition rotation $g_p \in \Gamma$, such that: $z_{p_-} = g_p z_{p_+}$. Since σ is continuous on $\mathcal{M}^c$, we have: $\sigma_{p_-} = \sigma_{p_+}$. Therefore, a tangent vector $X \in T_p\mathcal{M}$ has two limit images by the differential map: $\mathrm{d}\phi_{p_+}(X_p) = z_{p_+}\sigma_p(X_p)$ and $\mathrm{d}\phi_{p_-}(X_p) = z_{p_-}\sigma_p(X_p)$. Using the continuity of σ, we have:

$$\mathrm{d}\phi_{p_+}(X_p) = g_p \mathrm{d}\phi_{p_-}(X_p), \quad g_p \in \Gamma.$$

Therefore, vectors tangent to the cut are mapped to vector equal up to a rotation defined by the frame field discontinuity. □

4.2.3 Parametrization Design

Following Cartan's method, computing a parametrization is equivalent to finding local frames z with rotation speed ω and a 1-form σ satisfying the structure equations. Thus, we can setup an optimization problem aiming at finding z, ω, σ minimizing a distortion energy $\mathcal{D}$ and satisfying Equations (4.2), (4.3) and (4.4):

$$\begin{array}{lll} \min & \mathcal{D}(\sigma) & \\ z : \mathcal{M} \to \mathbb{U}/\Gamma & & \mathrm{d}z - \imath\omega z = 0 \\ \omega : \mathcal{M} \to \mathbb{R} & \text{s.t.} & \mathrm{d}\sigma + \imath\omega \wedge \sigma = 0 \\ \sigma : T\mathcal{M} \to \mathbb{C} & & \star\mathrm{Im}(\bar{\sigma} \wedge \sigma) > 0 \end{array} \tag{4.6}$$

The optimization constraints ensure the existence of a global parametrization of the cut surface M^c but never use the position of the cuts or the singular points. In fact, cones are determined during optimization and cuts are discovered during the construction of the parametrization when solving Equation (4.1). This formulation allows for a great range of possible constraints: boundary constraints, feature constraints, distortion measure, singularity positions as-well-as extensive freedom to let the cone be placed according to the distortion minimization.

Feasible region In theory, for a mapping $\phi : \mathcal{M}^c \to \mathbb{C}$, there are an infinite number of pairs z, σ satisfying Thm. 4.2. In particular, one can rotate the frame and the deformation tensor at each point by a rotation $r_p \in \mathbb{U}$ such that $\bar{z}_p := r_p z_p, \bar{\sigma}_p := r_p^{-1}\sigma_p$ does not alter the deformation — i.e. $\mathrm{d}\phi_p = \bar{z}_p\bar{\sigma}_p$. In order to satisfy Equation (4.2) and Equation (4.3), the rotation speed $\bar{\omega}$ of $\bar{z}$ must change according to the formula $\bar{\omega} = \omega - \imath r^{-1}\mathrm{d}r$. Thus, for any rotation field, if z, ω, σ are feasible then $\bar{z}, \bar{\omega}, \bar{\sigma}$ are also feasible with possibly the same distortion energy. Therefore, the feasible region in Equation (4.6) is extremely large and the optimization problem is highly non-convex. In practice, the size of the feasible set allows us to easily converge to a feasible solution (except for some specific configurations discussed in Sec. 4.9). However, we cannot guarantee that we will find a global minimum of the distortion energy.

4.2.4 Overview

Our goal is to discretize the optimization problem in Equation (4.6). We will first recall definitions necessary for discrete parametrization in Sec. 4.3. The constraints on frame rotations will be the topic of Sec. 4.4 and the first structure equation will be studied in Sec. 4.5. We will show in Sec. 4.6 that the discrete constraints are equivalent to the existence of a discrete parametrization. Sec. 4.7 discusses boundary conditions and feature alignment. Sec. 4.8 and Sec. 4.9 describe our numerical optimization scheme as-well-as our results and applications.

4.3 Discrete Setting

In this section, we recall the properties given in Section 2.4.2 of Chapter 2 we are trying to enforce on our resulting parametrization.

In the discrete setting, the mappings we consider are *piecewise linear parametrizations*, that is maps $\phi : \mathcal{M} \subset \mathbb{R}^3 \to \mathbb{C}$ assigning, to every triangle corner $c \in \mathbb{R}^3$ a coordinate in the (complex) plane $\phi_t(c) = (u, v)$. Coordinates for points inside a triangle are then linearly interpolated from its corners. Such a parametrization can exhibit *cuts* along some edges, namely adjacent triangles in the initial mesh are no longer adjacent in the parametrization. Linear *transition functions* g relate an edge on one side of the cut to its duplicated version on the other side. Given a closed loop of triangles $(t_0, t_1, \ldots, t_p, t_0)$, all incident to the same vertex $i \in V$, the vertex is said *singular* if the accumulated transition function $g_i = g_{t_0 \to t_p} \circ \ldots \circ g_{t_2 \to t_1} \circ g_{t_1 \to t_0}$ is not identity.

We will require in addition the parametrization to be *rotationally seamless*, meaning that duplicated edge vectors are equal up to a $k\pi/2$ rotation for any two adjacent triangles (Definition 2.2). More generally, we consider parametrization with quantized cones, for which transition functions are rotations of angle $2k\pi/n$ for $n \in \mathbb{N}^+$.

Another important property we consider here is that boundary curves and feature edges are isolines of the parametrization (Definition 2.3).

4.3.1 The Cayley Map

During the process of discretizing the integrability equations, we will see that rotations are represented using the *Cayley map* [Kobilarov et al. 2009]. By definition this map, noted $cay : \mathbb{R} \to \mathbb{U}$, associates to a real value $\alpha \in \mathbb{R}$ a complex fraction equal to the rotation of angle $2\arctan(\alpha/2) \in (-\pi, \pi)$:

$$cay(\alpha) := \frac{1 - \imath\alpha/2}{1 + \imath\alpha/2} = \exp\left(2\imath \arctan \frac{\alpha}{2}\right). \tag{4.7}$$

The Cayley map is often used as an efficient way of parametrizing the space of rotations by a polynomial function [Zhang et al. 2021]. Moreover, it defines a single covering of the space of rotations which is a desirable property for numerical optimization.

4.4 Local Frames

In this section we propose a discretization of our first integrability equation Equation (4.2). To do so, we need to define our local frames and how to compare them on a curved surface using a parallel transport. As we consider tangent vectors defined at vertices, the material exposed in Sec. 4.4.2 is inspired by Knöppel et al. [2013].

4.4.1 Vertex-Based Frame Fields and Singular Triangles

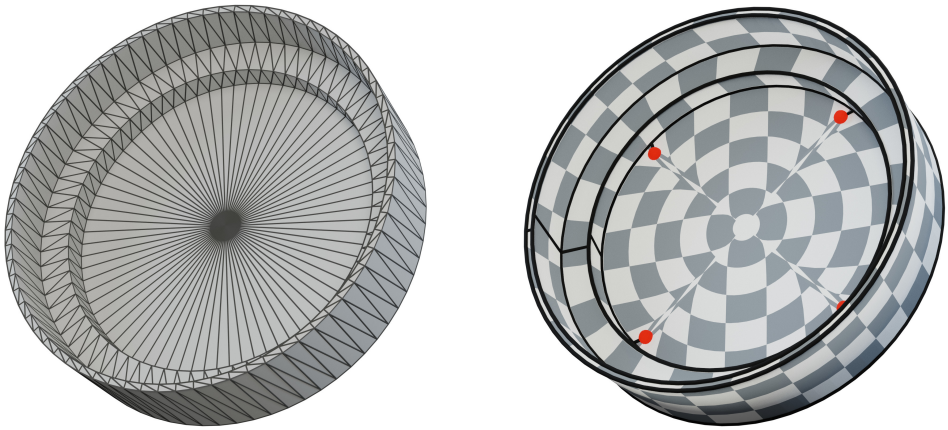


Figure 4.3: Sampling local frames at vertices creates cones inside triangles. Our algorithm adds extra vertices inside singular triangles allowing us to parametrize extremely coarse surfaces. In comparison, a face-based method is unable to create a valid parametrization due to the lack of degrees of freedom.

The choice of vertex-based frame fields can come as unusual in the context of cone parametrization. Indeed, the more common choice of frames on faces defines singularities at vertices and cuts along edges which is, in practice, very convenient. However, frames on vertices give more room for singularity placement as a cone may appear anywhere inside a triangle. Therefore, our algorithm succeeds in parametrizing poorly sampled surfaces by adding a new vertex inside each singular triangle as demonstrated in Figure 4.3. Such parametrizations are inaccessible to face-based algorithms.

Frames on faces are also easy to constrain at boundaries and at feature curves. Again, these constraints are also easily accessible for vertex-based field by adapting the flattening of the local vertex chart, as we will see in Sec. 4.7.

Most importantly, we can prove that our framework has to create cones of same index as the frame field. This is not guaranteed in the face-based paradigm due to possible creation of unwanted "double covering" [Garanzha et al. 2022]. In any case, a face-based discretization of our integrability equations is also possible.

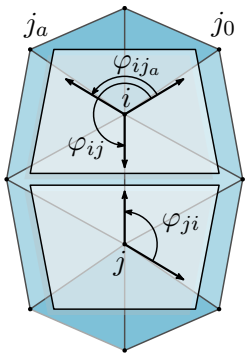
4.4.2 Parallel Transport From Levi-Civita connection

Our construction of a vertex-based frame field follows the one by Knöppel et al. [2013]. A tangent vector $v_i \in \mathbb{C}$ is expressed in the local basis of the vertex i. This local basis defines a *tangent space* at this vertex. Vertices on a triangulated surface are generally not flat as the inner angles θ_i^{jk} of triangles incident to i do not sum to 2π. A local coordinate system is constructed by intrinsically "flattening" each vertex, namely inner angles are normalized:

$$\tilde{\theta}_i^{jk} := 2\pi\theta_i^{jk}/\Theta_i$$

where $\Theta_i = \sum_{ijk} \theta_i^{jk}$ is the total inner angle at vertex i. At each vertex, we assign a reference edge ij_0 whose angle coordinate φ_{ij_0} is by definition zero. The angles of other ordered edges $ij_0, \dots, ij_n$ are obtained by accumulating modified inner angles:

$$\varphi_{ij_a} := \sum_{p=0}^{a-1} \tilde{\theta}_i^{j_p j_{p+1}}. \tag{4.8}$$



To compare adjacent vectors, we define the parallel transport $\rho : E \to \mathbb{R}$ as the rotation angle aligning the basis at j to the one at i:

$$\rho_{ij} := \varphi_{ji} - \varphi_{ij} + \pi,$$

by comparing the angles of the shared edge ij. The change of basis is then $r_{ij} = \exp(\imath\rho_{ij})$. By construction, this parallel transport is associated to the Levi-Civita connection. As shown by Knöppel et al. [2013], its integrated Gaussian curvature $K \in (-\pi, \pi]$ defined as

$$\exp(\imath K_{ijk}) = r_{ki} r_{jk} r_{ij}$$

satisfies the Gauss-Bonnet theorem (for meshes without boundaries).

4.4.3 Discrete Frame Field Condition

We are now ready to discretize the frame equation Equation (4.2) using Discrete Exterior Calculus [Hirani 2003].

A frame field is an assignment of unit complex number z per vertex. So, the exterior derivative of the frame is simply the frames difference corrected by the parallel transport:

$$(\mathrm{d}z)_{ij} = z_j - r_{ij} z_i.$$

For the second part of the equation, we integrate a wedge product between a 1-form and a 0-form along a primal edge following Hirani [2003]; Ptáčková and Velho [2021]:

$$(\omega z)_{ij} = \imath\omega_{ij}(z_j + r_{ij} z_i)/2.$$

Combining these two equations, the discrete frame constraint Equation (4.2) reads:

$$(1 + \imath\omega_{ij}/2)r_{ij}z_i = (1 - \imath\omega_{ij}/2)z_j. \tag{4.9}$$

Equation (4.9) implies that the frame at vertex i is equal to the frame at vertex j up to a multiplication by $cay(\omega_{ij})$. So adjacent frames are equal up to a rotation of angle $2\arctan(\omega_{ij}/2)$. This is precisely the intuition behind the continuous equation.

4.4.4 Quantized Singularity Cones Fields

According to Thm. 4.2, we can define a cone parametrization if the local bases admit quantized cone singularities. This property is satisfied exactly by discrete symmetric frame fields [Ray et al. 2008].

A rotationally symmetric n-direction field (n-RoSy field [Vaxman et al. 2016]) assigns at each vertex a set of n directions:

$$\{\exp(\imath 2k\pi/n)\, z_i, \quad k = 0, \ldots, n\}\,.$$

A key insight is that, when raised to the n^{th} power, this set reduces to a single complex value:

$$v_i := z_i^n.$$

Thus, it can be uniquely represented by a unit complex number whose n^{th} roots gives the n directions [Palacios and Zhang 2007; Ray et al. 2008]. For v to represent a n-RoSy direction field, we simply raise Equation (4.9) to the power n:

$$\Big|\ (1 + \imath\omega_{ij}/2)^n v_j \quad = (1 - \imath\omega_{ij}/2)^n r_{ij}^n v_i. \tag{$\mathcal{F}_n$}$$

The symmetric field may be smooth over the surface, it can still represent a discontinuous frame field. Taking the n^{th} root of Equation ($\mathcal{F}_n$), we can quantify the rotation jumps in the frames by an integer $k_{ij} \in \mathbb{Z}$ per edge:

$$z_j = \exp\left(2\imath\pi k_{ij}/n\right) cay(\omega_{ij}) r_{ij} z_i.$$

We will see with Theorem 4.3 that a rotation jump by an angle $2\pi k_{ij}/n$ in the frame field will create a cut with the same transition functions in the parametrization.

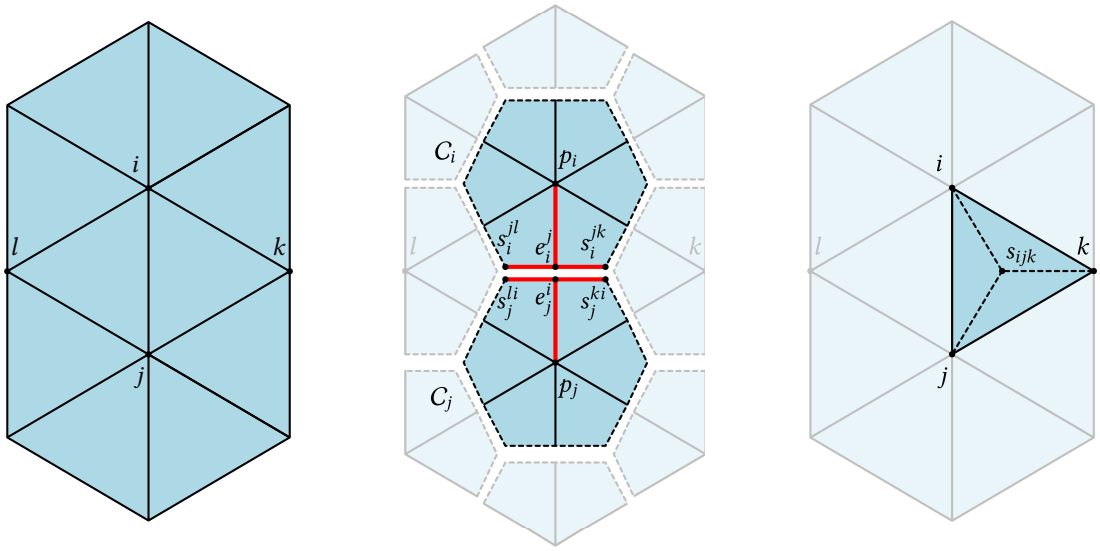
4.5 First Structure Equation

We are now ready to discretize the other equation needed for integrability a.k.a. the first structure equation Equation (4.3).

4.5.1 Vertex Charts

The 1-form σ expresses a deformation of the mesh in the coordinates of a local frame. Thus, we need to define a neighborhood or *chart* affected by this local basis.

As singularities may appear inside triangles, the charts must include a point inside each triangle physically representing the (potential) cone. For this reason, we build our charts as the intersection of a triangle mesh M with its dual. Namely, each triangle is split into three quads, as in Figure 4.4b, by adding three edges linking the triangle center to edge midpoints. We explicitly build the chart $\mathcal{C}_i$ at vertex i as the union of quads containing i. In this construction, adjacent



(a) M (b) M^c (c) M_p

Figure 4.4: The initial mesh (a) is subdivided in a chart collection (b) used for imposing integrability constraints. Edge midpoints are denoted with letter e and triangle center point with the letter s. We do not parametrize the initial mesh but a subdivision of it where singularities s are inserted inside singular triangles (c).

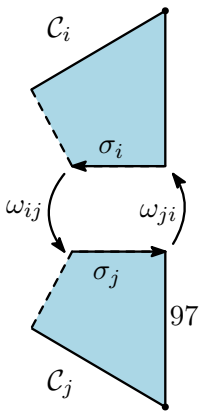
charts are disjoint but share two types of duplicated edges: primal edges from the input mesh and dual edges linking edge centers to triangle centers. Both types of edges are highlighted in red in Figure 4.4b. We define the chart collection $M^c = \cup_{i \in V} \mathcal{C}_i$ as the set of all charts emanating from the subdivision of the input mesh. Effectively, we will parametrize the mesh M_p exactly equal to the initial mesh except at singular triangles which will be split into three to insert singular points (Figure 4.4c).

In the DEC formalism, 1-forms are discretized using their integrated values along edges. By definition, the deformation of an edge vector u of M is given by $\sigma(u) \in \mathbb{C}$. To ease the notations, we directly use the deformed edges as variables by using specific letters for each point type of the deformed mesh. The letter p denotes the vertex coordinates (with a complex number) of the input mesh in parameter space, the letter e is the midpoint of an edge and s the center of the triangle and thus a potential singularity. Moreover, under-scripts are reserved to make explicit the membership of a vector to a chart. For example, $s_i^{jk} - e_i^j$ is a deformed dual edge in the chart $\mathcal{C}_i$ whereas $s_j^{ki} - e_j^i$ is its copy in the chart $\mathcal{C}_j$.

4.5.2 Discrete Structure Equation

In order to discretize Equation (4.3), we use Discrete Exterior Calculus [Hirani 2003] on "lens" complexes as introduced by Soliman et al. [2021]. A lens complex is obtained by gluing opposite halfedges of adjacent charts. Note that σ_i is expressed in the basis z_i, thus, in order to compare two 1-forms living in adjacent charts, we must compensate for the basis rotation by multiplying by the *inverse* parallel transport. So, along a dual edge, the integral exterior derivative reads:

$$(\mathrm{d}\sigma)_{ij} := r_{ji}(s_i^{jk} - e_i^j) - (s_j^{ki} - e_j^i).$$



The wedge product between a primal 1-form ω and a lens complex can be thought of as a wedge product on a virtual quadrangle made of primal and dual edges where the frame rotations ω on dual edges are zero. Using the wedge product on a quadrilateral element defined by Krantz and Parks [2008]; Ptáčková and Velho [2021], we obtain:

$$(\omega \wedge \sigma)_{ij} = \imath\omega_{ij}(-r_{ji}(s_i^{jk} - e_i^j) - (s_j^{ki} - e_j^i))/2.$$

When applied to both the primal and dual edges of a chart, we obtain the system of equations:

$$\left| \begin{array}{lcll} (1 + \imath\omega_{ij}/2)\left(e_j^i - p_j\right) & = - & (1 - \imath\omega_{ij}/2)\, r_{ji}\left(e_i^j - p_i\right), & \forall ij \in E \\ (1 + \imath\omega_{ij}/2)\left(s_j^{ki} - e_j^i\right) & = & (1 - \imath\omega_{ij}/2)\, r_{ji}\left(s_i^{jk} - e_i^j\right). & \forall ijk \in F \end{array} \right. \tag{$\mathcal{E}$}$$

Again, we notice that any edges shared by two adjacent charts are equal up to a rotation but this time it involves the inverse Cayley transform $cay(\omega_{ij})^{-1}$ and the inverse parallel transport.

Note that Equations ($\mathcal{E}$) are written only in term of the chart edges. So, without loss of generality, we can reduce the number of variables by fixing the charts translations and setting $p_i = 0$, $\forall i \in V$.

4.5.3 Local Injectivity

To obtain a valid parametrization, we also need to ensure local injectivity. In other words, triangle areas must remain positive in parameter space. This boils down to two different constraints. First, the orientation of primal edges in vertex charts should be preserved:

$$\left| \det(p_i - e_i^j, p_i - e_i^k) > 0. \right. \tag{$\mathcal{I}_1$}$$

Second, the singularity point s_{ijk} should stay inside triangle ijk:

$$\left| \begin{array}{ll} \det(p_i - e_i^j, p_i - s_i^{jk}) & > 0, \\ \det(p_i - s_i^{jk}, p_i - e_i^k) & > 0. \end{array} \right. \tag{$\mathcal{I}_2$}$$

4.6 Quantized Cone Parametrization

Now that we have all integrability conditions in hand, we are ready to reconstruct the parametrization from local bases and chart coordinates. Furthermore, assuming that the charts satisfy the system of equations in Equations ($\mathcal{E}$) and Equation ($\mathcal{F}_n$), we can show that it defines a valid quantized cone parametrization.

4.6.1 Parametrization Reconstruction

Recovering a parametrization from the chart collection M^c, is very simple as we just need to integrate Equation (4.1). This equation states that parametrized edges are simply the chart edges but rotated by the basis z:

$$(\mathrm{d}\phi)_i = z_i\sigma_i.$$

Since edges of two adjacent charts $\mathcal{C}_i, \mathcal{C}_j$ and two adjacent frames z_i, z_j are both equal up to a rotation of $cay(\omega_{ij})r_{ij}$, the rotated charts $z_i\mathcal{C}_i, z_j\mathcal{C}_j$ can be stitched together by finding a

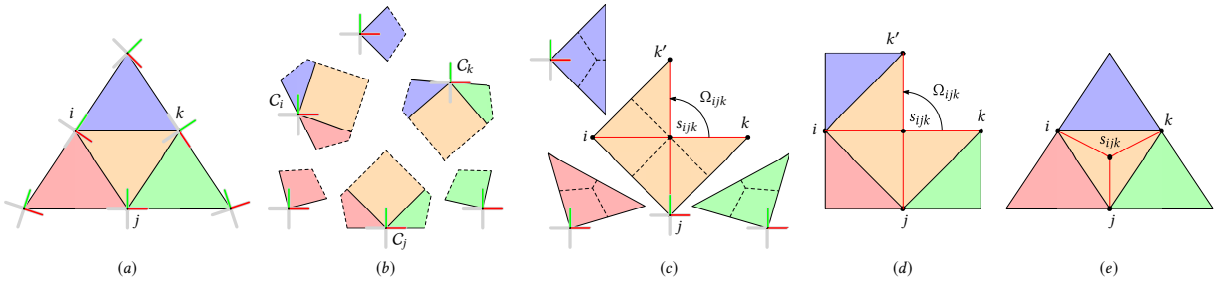


Figure 4.5: Reconstruction of a parametrization from frames and charts. A triangle mesh (a) is divided into a set of vertex charts (b) satisfying the integrability conditions of Equations ($\mathcal{E}$), namely duplicated edges are equal up to the cross field rotation $cay(\omega)^{-1}$. The triangles are assembled independently (c). The triangle ijk is singular so the vertex s_{ijk} is added along with new edges in red. The angle Ω_{ijk} at the cut matches the discontinuity of the frame field. The parametrization (d), obtained by gluing adjacent triangles, corresponds to the mesh (e) where the singular point is inserted in triangle ijk using its barycentric coordinates in parameter space.

translation. However, when using a reconstruction based on vertex charts, singular triangles will be split into three quads and parametrization cuts will follow dual edges. Thus, in order to minimize connectivity changes, it is better to first reconstruct triangles from quads and then build the parametrization from triangles.

To do so, we need to recover the local bases z from their power representation v. As shown by Thm. 4.2, the discontinuities of the frame field determine the position of the parametrization cuts. Since there are n vectors satisfying $v = z^n$, the choice of the root of unity should not be random. Thus, we compute the bases along a spanning tree on quads prioritizing coherent bases on triangles so that cuts will appear on primal edges and changes in mesh topology will be minimal.

Then, we rotate the quads according to their bases z and we reconstruct all triangles independently by applying the appropriate translation to each quad. Singular triangles are triangulated by adding edges linking the singular point s and triangle vertices. However, if the singularity position is known in parameter space, its position on the surface mesh is not yet determined. To overcome this issue, we place the point s_{ijk} on the surface using the barycentric coordinates in parameter space.

The parametrization is then reconstructed by recursively stitching triangles along the spanning tree. This guarantees that cuts only appear at primal edges except for singular triangles who are split into three quads.

In summary, the parametrization is computed by following the steps:

1. Recover a basis z per quads from the power vector v by finding the smallest symmetry rotation along a tree;

2. Rotate each quad with their respective basis;

3. Reconstruct triangles and remesh singular ones;

4. Recursively stitch adjacent triangles along the tree.

This reconstruction process is summarized in Figure 4.5.

4.6.2 Theoretical Guarantees

Crucially, we can prove that if Equations $(\mathcal{E})$, $(\mathcal{I}_1)$, $(\mathcal{F}_n)$ and $(\mathcal{I}_2)$ are satisfied, the reconstruction leads to a valid quantized parametrization whose transition functions are equal to the rotational discontinuity of the frame field z. Figure 4.1 shows an example of parametrization obtained by solving Cartan's structure equation.

Theorem 4.3. *If a set of charts, a frame field and its rotations satisfy the system of Equations $(\mathcal{E})$, $(\mathcal{F}_n)$, $(\mathcal{I}_1)$ and $(\mathcal{I}_2)$, then we can recover a locally injective parametrization with quantized cones whose transition functions are rotation equal to the frame field discontinuities.*

Proof. In order to show that our constraints define a valid parametrization with quantized cone points in the sense of Def. 2.2. We will follow four steps. 1) We will show the validity of the triangle reconstruction (Sec. 4.6) and that transition functions are rotations of angle $2k\pi/n$. 2) The parametrization is locally injective. 3) The parametrization's angle defects are equal to the singular indices of the frame field up to a *subtraction* of an integer multiple of 2π. 4) Due to the Gauss-Bonnet theorem, parasite singularities cannot appear.

Triangle reconstruction. First, we will show that the parametrization cuts appear at the frame field discontinuities and that the rotation jumps match those of the frames. Since the power field satisfies Equation $(\mathcal{F}_n)$, the bases z rotations are given by the formula:

$$z_j = \exp\left(2\imath\pi k_{ij}/n\right) cay(\omega_{ij}) r_{ij} z_i,$$

where k_{ij} is the jump index at edge ij.

Let $\sigma_i \in \mathbb{C}, \sigma_j \in \mathbb{C}$ be duplicated *primal* edge vectors of adjacent charts $\mathcal{C}_i, \mathcal{C}_j$ and $a = z_i\sigma_i, b = z_j\sigma_j$ their coordinates in parameter space. Using Equations $(\mathcal{E})$ and Equation (4.9), we see that they are opposite in parameter space:

$$\begin{aligned} b &= z_j\sigma_j, \\ &= \left(\exp\left(2\imath\pi k_{ij}/n\right) cay(\omega_{ij}) r_{ij} z_i\right) \left(-\, cay(\omega_{ij})^{-1} r_{ij}^{-1} \sigma_i\right), \\ &= -\exp\left(2\imath\pi k_{ij}/n\right) z_i\sigma_i, \\ &= -\exp\left(2\imath\pi k_{ij}/n\right) a. \end{aligned}$$

The same reasoning can be be applied to *dual* edges. Thus, primal and dual edges are equal up to the frame rotational discontinuity $d_{ij} := \exp\left(2\imath\pi k_{ij}/n\right)$. Therefore, our parametrization transition functions are rotation quantized by the frame field. In particular, our parametrization is seamless whenever $n = 4$.

To achieve our triangle reconstruction of the parametrization, three cases can occur. 1) A triangle is not traversed by a cut and it will be reconstructed as a triangle in the parametrization. 2) A *non-singular* triangle is traversed by a cut, thus the cut can be displaced on primal edges to recover a non-singular triangle. 3) A *singular* triangle is reconstructed with a cut appearing in one of the dual edge linking the point s. By displacing the cut, we recover a triangulation of the singular triangle with the cut linking s to one of the vertex as in Figure 4.5.

Injectivity and feature constraints. Equations $(\mathcal{I}_1)$ ensure local injectivity as all triangles (p_i, p_j, s_{ijk}) composing the parametrization must have positive area.

Parametrization angle defects. We are now going to precisely relate the parametrization angle defects with the frame singular indices. We showed that the parametrization transition functions match the frame field jumps. Therefore, the parametrization angle defect at a singular point is

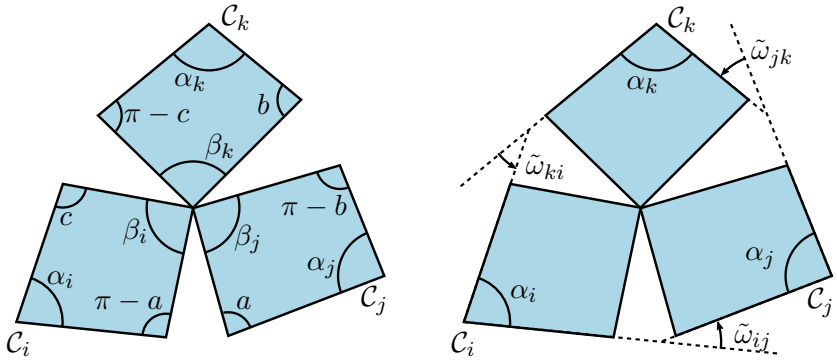


Figure 4.6: The triangle ijk is decomposed into the three charts $\mathcal{C}_i, \mathcal{C}_j$ and $\mathcal{C}_k$ satisfying Equations $(\mathcal{E})$. Left: the angles α denote the triangle angle and β the angle at the singularity point. Right: primal edges of the charts are equal up to the rotation $cay(\omega)$ whose angle is $\tilde{\omega} = 2\arctan(\omega/2)$ (see Equation (4.7)).

equal to the total rotation angle of the frame up to an integer multiple of 2π. We will show that this additional parasite angle corresponding to integer singularities can only be negative.

Let us consider a triangle ijk and its chart decomposition as illustrated by Figure 4.6. Let Ω_{ijk} be the frame field total rotation around the triangle, that is to say the singularity index prescribed by the frame field:

$$\Omega_{ijk} = \tilde{\omega}_{ij} + \tilde{\omega}_{jk} + \tilde{\omega}_{ki},$$

where $\tilde{\omega} = 2\arctan(\omega/2)$ are the angle of the rotation between two frames, parametrized by the Cayley transform (Equation (4.7)).

Let Θ_{ijk} be the angle defect of triangle ijk in parameter space, by definition:

$$\Theta_{ijk} = 2\pi - (\beta_i + \beta_j + \beta_k)$$

Because of constraints $(\mathcal{I}_1)$ and $(\mathcal{I}_2)$, the three charts $\mathcal{C}_i, \mathcal{C}_j$ and $\mathcal{C}_k$, decomposing triangle ijk, are non self-intersecting quadrilaterals. Thus, their four inner angles, defined in Figure 4.6 (left), must sum to 2π:

$$\begin{aligned}
\alpha_i + \beta_i + c + \pi - a &= 2\pi, \\
\alpha_j + \beta_j + a + \pi - b &= 2\pi, \\
\alpha_k + \beta_k + b + \pi - c &= 2\pi.
\end{aligned}$$

Moreover, by summing these three equations, we obtain a link between the angle defect Θ and the triangle angles α:

$$\Theta_{ijk} = \alpha_i + \alpha_j + \alpha_k - \pi.$$

We are now ready to relate Θ with the frame field singularity angle Ω. Let us consider the hexagon formed by extending primal edges of charts until intersection, as in Figure 4.6 (right). This hexagon can be self-intersecting but remains star-shaped around its singularity point s. We can express the sum of its inner angles in term of its turning number $k > 0$:

$$\alpha_i + \alpha_j + \alpha_k + 3\pi - \tilde{\omega}_{ij} - \tilde{\omega}_{jk} - \tilde{\omega}_{ki} = \pi(6 - 2k).$$

Rearranging the terms, the parametrization angle defect Θ is equal to the frame field total rotation Ω up to a negative integer multiple of 2π:

$$\Theta_{ijk} = \Omega_{ijk} - 2\pi k, \quad k \in \mathbb{N}. \tag{4.10}$$

Double coverings. We are now going to prove that the parametrization singularities are exactly those prescribed by the frames. Equation (4.10) shows that inside triangles the parametrization angle defect can differ from the frame field singularities by a negative integer multiple of 2π. Vertices of the input mesh also have prescribed singularity indices: 0 for inner vertices and user prescribed on feature edges. At these vertices, unwanted *negative* integer index singularities could also appear even if all triangles have positive area (see Figure 4.7).

However, due to the Gauss-Bonnet theorem, an additional *negative* singularity can appear only if it is compensated by an additional *positive* singularity. This is impossible because only *negative* parasite singularities can appear at both triangles and vertices.

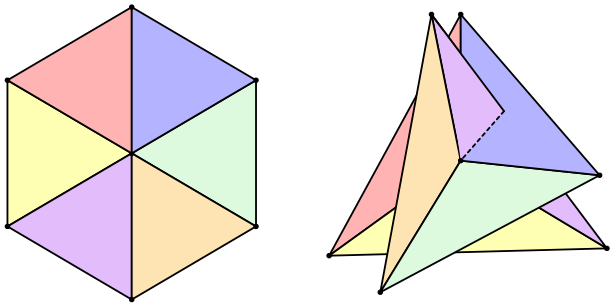


Figure 4.7: Two versions of the same vertex rings: one with an angle defect of 0 and the other with an angle defect of 2π, both have only positive area triangles.

□

4.7 Boundary and Feature Edge Adaptation

Another requirement we discussed in Section 2.4.2 is that the parametrization should be *adapted* to the mesh boundary and features edges (Fig 2.21c). Additional constraints are therefore required on the cross field and on vertex charts. We process feature edges extracted from a boundary or user specified in the same way. We assume that the user provides us, at each vertex of a feature curve, with the target angle $\Omega_i = 2\pi k_i/n$ in parameter space.

Single vertex corner We define a vertex corner as a set of adjacent triangles incident to the same vertex i and delimited by two feature edges ij_0 and ij_a. In this paragraph, we consider a single vertex corner. We assume that target corner angles $\Omega_i^{j_0 j_a}$ integer multiple of $2\pi/n$ are provided as input. At a corner ij_0j_a, inner angles are normalized to match the prescribed angle:

$$\tilde{\theta}_i^{j_p j_{p+1}} := \left(\Omega_i^{j_0 j_a} / \Theta_i^{j_0 j_a}\right) \theta_i^{j_p j_{p+1}}$$

where $\Theta_i^{j_0 j_a}$ is the total inner angle between edges ij_0 and ij_a. The edge angles in the tangent plane are obtained by accumulating the modified inner angles as in Eq. (4.8). The representation vector v_i is constrained to be equal to $e^{ni\varphi_{ij_0}}$ so that one of its root agrees by construction with the feature edge direction. An illustration of this procedure is given in Figure 4.8.

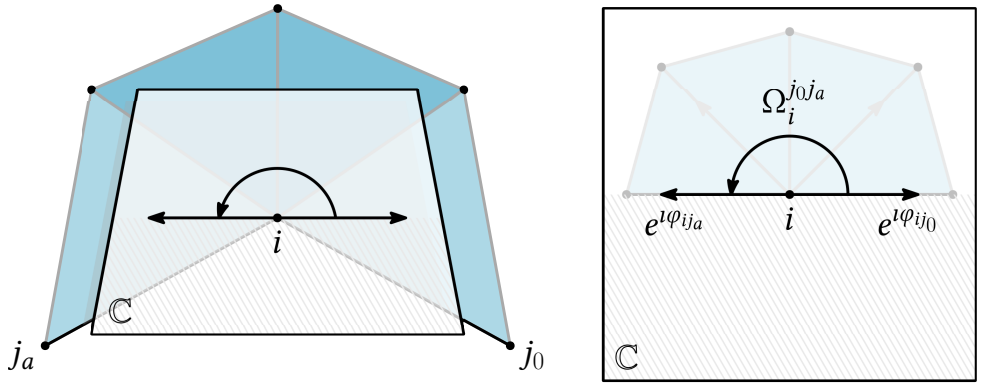


Figure 4.8: Boundary edges, depicted in black, are laid out in the tangent complex plane so that they match the provided corner angle $\Omega_i^{j_0 j_a}$.

The corresponding vertex corner in chart $\mathcal{C}_i$ must also be constrained to the corner angle. To do so, the primal edge vector corresponding to a feature edge must remain orthogonal to the feature edge normal. Therefore, at a vertex corner, we enforce three constraints linear with respect to the chart coordinates and the cross field:

$$\left|\begin{array}{l} v_i = \exp(n\imath\varphi_{ij_0}), \\ \langle p_i - e_i^{j_0}, \imath\exp(\imath\varphi_{ij_0})\rangle = 0, \\ \langle p_i - e_i^{j_a}, \imath\exp(\imath\varphi_{ij_a})\rangle = 0. \end{array}\right. \qquad (\mathcal{B})$$

Note that since the frame field is constrained on feature vertices, the rotation ω is also known along feature edges.

Singular vertices Let us consider the case where an *interior* vertex i is incident to multiple feature edges $(ij_{a_0}, ij_{a_1}, \ldots, ij_{a_n})$ forming $n+1$ corners. If the sum of all corner angles $\Omega_i = \sum_{p=0}^{n-1} \Omega_i^{j_{a_p} j_{a_{p+1}}} + \Omega_i^{j_{a_n} j_{a_0}}$ is an integer multiple of 2π, then the flattening of each corner creates a valid vertex chart. However, when $\Omega_i \neq 2k\pi$, vertex i is *singular* and an additional seam is necessary to lay out edges in the tangent plane. Thus, we cut open the vertex neighborhood at edge ij_{a_0} introducing a duplicated edge ij'_{a_0}. For the parametrization of chart $\mathcal{C}_i$ to remain valid, edges on each side of the cut must be equal in length, yielding an additional linear constraint:

$$\left| \langle p_i - e_i^{j_{a_0}}, \exp(\imath\varphi_{ij_{a_0}})\rangle = \langle p_i - e_i^{j'_{a_0}}, \exp(\imath\varphi_{ij'_{a_0}})\rangle. \right. \qquad (\mathcal{B})$$

Figure 4.9 illustrates the computation of the tangent plane at a cube corner where $\Omega_i = 3\pi/2$.

4.8 Numerical Optimization

So far, we have defined a system of necessary equations for computing valid global parametrizations. In this section, we describe our optimization process in order to compute the vertex charts, the frame field rotations ω and local bases z (represented, when necessary, by its power vector v) which are solutions of this system.

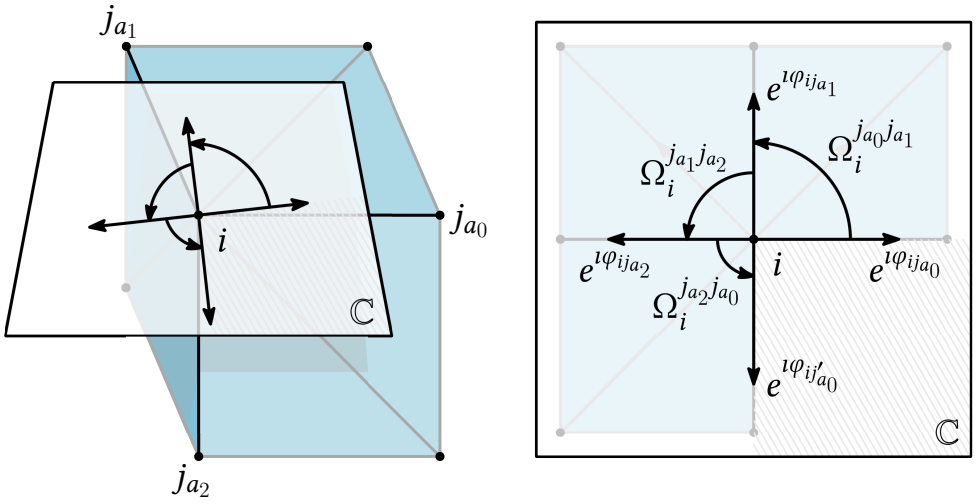


Figure 4.9: Three feature edges, depicted in black, are laid out in the tangent complex plane. A seam is introduced at edge ija_0 because the sum of the corner angles Ω is not an integer multiple of 2π.

4.8.1 Non-Linear Least-Squares

Like the smooth optimization problem of Equation (4.6), we would like to find the chart vertices (x, y), the frame rotation ω and the field power vector v solution of the constrained optimization problem:

$$\begin{aligned} \min_{\omega,v,x,y} \quad & \mathcal{D}(x,y) \\ \text{s.t.} \quad & (\mathcal{E}), (\mathcal{F}_n), (\mathcal{B}), (\mathcal{I}_1), (\mathcal{I}_2), \end{aligned} \tag{4.11}$$

where $\mathcal{D} : M^c \to \mathbb{R}$ is any given distortion energy. Note that to solve the optimization problem in Equation (4.11), we do not need to know in advance the position of the cones or the cuts; everything will be decided during optimization.

To simplify the problem, the inequality constraints of Equations $(\mathcal{I}_1)$ and $(\mathcal{I}_2)$ are enforced using the continuously differentiable barrier function $B_\eta : \mathbb{R}_{>0} \to \mathbb{R}$:

$$B_\eta(x) = \begin{cases} +\infty & x \leq 0, \\ \log(\frac{x}{\eta}) & 0 < x \leq \eta, \\ 0 & \eta \leq x. \end{cases}$$

Given a determinant d whose initial value is d_0, we set the threshold to $\eta = d_0/2$.

Furthermore, we enforce non-linear constraints as a sum of squares energy. Abusing the notations, let $\mathcal{E}$ and $\mathcal{F}_n$ denote the vectors formed by the differences of left and right hand sides in equations $(\mathcal{E})$ and $(\mathcal{F}_n)$. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be the vectors of determinants appearing in equations $(\mathcal{I}_1)$ and $(\mathcal{I}_2)$. We reformulate the constraints as the energy:

$$R(\omega, v, x, y) = \|\mathcal{E}\|_2^2 + \lambda_F \|\mathcal{F}_n\|_2^2 + \lambda_{\text{det}} \|B_\eta(\mathcal{I}_1)\|_2^2 + \lambda_{\text{det}} \|B_\eta(\mathcal{I}_2)\|_2^2,$$

where the function B_η is applied componentwise. This leads to a non-linear least-squares optimization with linear constraints:

$$\min_{\omega,v,x,y} \varepsilon\mathcal{D}(x,y) + R(\omega, v, x, y) \qquad \text{s.t. } (\mathcal{B}). \tag{4.12}$$

We use the Levenberg-Marquardt algorithm, an optimizer tailored for solving non-linear least squares with zero as global minimum, to minimize this objective function. Each optimization step consists in solving a quadratic optimization problem with linear constraints involving the function's Jacobian matrices. We follow the implementation of Marumo et al. [2020] whose evolution of the damping parameter ensures global convergence toward a local minimum and a second order rate of convergence near a zero of the objective function.

The problem in Equation (4.12) is solved several times for decreasing values of ε. For our distortion minimizing experiments, we start at $\varepsilon = 10^2$ and divide by a factor of 10 until $\varepsilon = 10^{-4}$, before solving one last time with $\varepsilon = 0$. This penalty scheme does not provide strong guarantee of convergence toward a global minimum of the distortion. However, in practice the choice of distortion energy greatly impacts singularity positions (see for example Figures 4.1 and 4.10) and distortion measure is indeed lower and competitive with other methods (Figures 4.20 or 4.13b).

Note that for surfaces with boundary or feature edges, we do not need to force v to be unit norm: it suffices that the frame field is constrained somewhere so that whenever Equations (4.9) are satisfied, the field has unit norm everywhere. For surfaces without features, we constrain v at a random vertex to avoid the trivial zero solution.

4.8.2 Distortion Energy

Many differentiable distortion metrics can be optimized to obtain interesting parametrizations. In our experiments, we consider three of these metrics.

As-Rigid-As-Possible distortion The first way to prevent charts distortion is to impose an isometric map between initial and final charts. This boils down to minimizing the so-called As-Rigid-As-Possible energy [Sorkine and Alexa 2007] which forces the map Jacobians to be orthogonal matrices:

$$\mathcal{D}_{\text{ARAP}}(x, y) = \sum_{i \in V} \sum_{t \in T_{\mathcal{C}_i}} \|J_t^T J_t - I_2\|^2$$

We refer to this energy as the *ARAP* or *isometric* energy.

Conformal distortion One important application of seamless parametrizations is quad remeshing. In this context, it is often desirable to avoid shearing *i.e.* quads with non-orthogonal edges as much as possible . The simplest way to promote these square-like quads is to penalize parametrizations whose parameter gradients are non-orthogonal. Thus, as a distortion energy, we use the *LSCM* [Lévy et al. 2002] energy summed over all charts:

$$\mathcal{D}_{\text{LSCM}}(x, y) = \sum_{i \in V} \sum_{t \in T_{\mathcal{C}_i}} \|\nabla y_t - \mathcal{R} \nabla x_t\|^2,$$

where (x, y) are the chart vertex coordinates and $\mathcal{R}$ is the $90°$ counter-clockwise rotation around the normal. The gradient operator is computed with respect to the initial isometric chart parametrization described in Sec. 4.8.3.

Area distortion Finally, one can require that the quad elements present the same overall size everywhere, which can be achieved by penalizing the change in chart area. This boils down to a constraint on the determinants of the Jacobian matrices, which should be as close as possible to 1. However, this class of area preserving deformations is not sufficient on its own as it also

contains solutions with dramatic shearing. We therefore balance our area distortion term with the previously defined $\mathcal{D}_{\text{LSCM}}$:

$$\mathcal{D}_{\text{AREA}}(x, y) = \sum_{i \in V} \sum_{t \in T_{\mathcal{C}_i}} \| \det(J_t) - 1 \|^2 + \mu \mathcal{D}_{\text{LSCM}}(x, y)$$

We set μ to 0.1 in our experiments.

Metrics comparison Figure 4.1 illustrates the influence of $\mathcal{D}_{\text{ARAP}}$, $\mathcal{D}_{\text{LSCM}}$ and $\mathcal{D}_{\text{AREA}}$ on cone positions and overall parametrization geometry. We quantitatively compare the resulting parametrizations using three criteria: the number of cones, the *scale* and the *stretch* distortion.

The *scale* distortion, defined over each triangle as $(\det J + \det J^{-1})/2$, quantifies how much triangles' area changes between initial geometry and parameter space. The *stretch* distortion, defined as the ratio σ_2/σ_1 of the sorted singular values ($\sigma_2 > \sigma_1$) of the Jacobian matrices, quantifies the deformation of angles. Both metrics have an optimal value of 1. We usually present them as a integrated quantity over the entire model.

In Fig 4.13b, we run our algorithm with distortion minimization on a large dataset of models and plot the resulting *scale* distortion against *stretch* distortion. As expected, $\mathcal{D}_{\text{LSCM}}$ reduces the stretch distortion, at a cost of a greater variation on the scaling of triangles thus producing parametrizations that are more conformal. On the other hand, minimizing $\mathcal{D}_{\text{AREA}}$ leads to parametrizations with little to no scale distortion. $\mathcal{D}_{\text{ARAP}}$ yields a good compromise between the previous two.

4.8.3 Initialization

As for any non-convex optimization, the question of a good initialization arises and initial values for our variables x, y, ω and v should be chosen carefully.

Charts. The charts coordinates are initialized using the tangent plane flattening introduced in Secs. 4.4 and 4.7. Given the normalized triangle inner angles, the chart primal edges are laid out in the plane with their initial edge lengths. The chart is completed by initializing points s at the triangle barycenters. These initial charts keep the triangle edge lengths unchanged so they can be seen as an *isometric* parametrization of the input triangle mesh. Since the system is invariant by chart translation, the central vertex is set to zero and all coordinates in a chart are given in relation to the origin.

Rotations. For surfaces without boundaries or feature edges, the frame rotations ω are simply initialized to zero. In some cases, this initialization can lead to highly distorted solutions. Typically, for CAD models where feature edges meet at very acute angles, a singularity appears directly inside the sharp triangle. Inspired by Desobry et al. [2022], we compute an initial frame rotation ω by forcing the cross field to be regular on all triangles $T(p)$ belonging to 3-ring of a sharp corner p. In practice, we solve the quadratic problem:

$$\begin{aligned} \min_{\omega \in \mathbb{R}^{|E|}} \quad & \textstyle\sum_{ij \in E} \|\omega_{ij}\|_2^2 \\ \text{s.t.} \quad & \omega_{ij} + \omega_{jk} + \omega_{ki} = -K_{ijk}, \quad \forall ijk \in T(p) \\ & \omega_{ij} \text{ fixed on feature edges} \end{aligned}$$

Figure 4.11 illustrates how this initialization drastically changes the parametrization and reduces the distortion.

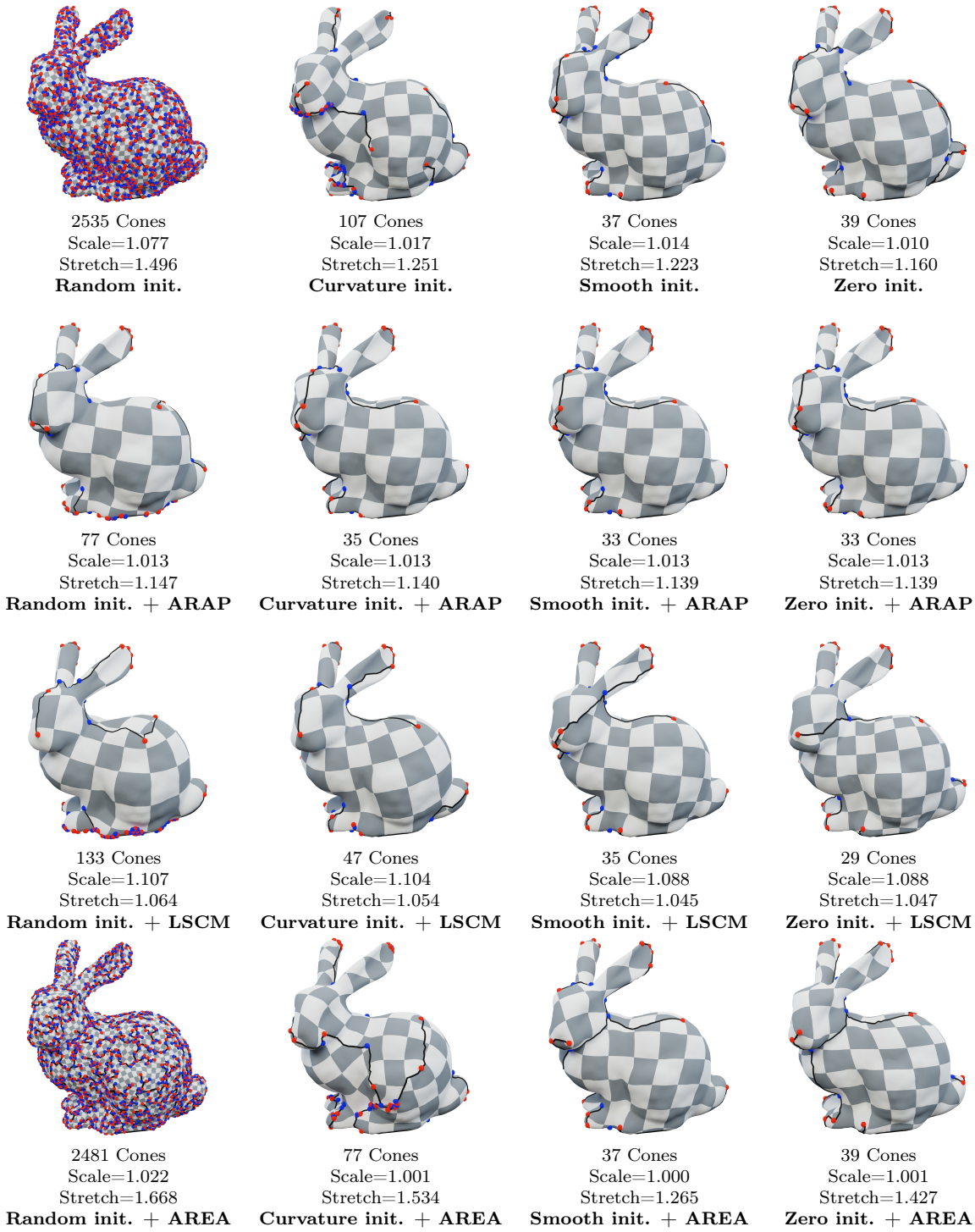


Figure 4.10: Parametrization of the Stanford Bunny with different frame field initializations. From left to right: random frame per vertex, principal curvature direction, smooth frame field [Viertel and Osting 2018] and zero except on boundary vertices. Rows are optimized respectively : without distortion, along $\mathcal{D}_{\text{ARAP}}$, along $\mathcal{D}_{\text{LSCM}}$ and along $\mathcal{D}_{\text{AREA}}$.

Frame field. Many different frame field initializations are possible as long as they respect boundary and feature edges constraints (see Sec. 4.7). In Figure 4.10, we compare four initializations: a random cross field, the principal curvature directions obtained with [Cohen-Steiner and Morvan 2003], the smoothest cross field computed *via* [Viertel and Osting 2018] or zero everywhere except at boundary and feature vertices. When no distortion is minimized, the algorithm converges to the singularity configuration specified by the cross field which is the closest valid parametrization. On the other hand, any reasonable initialization will allow our algorithm to converge to a very similar cone distribution as soon as a distortion energy is minimized. This is due to the fact that the distortion energy dominates the optimization in early iterations and guides the charts towards finding the position minimizing it. The cross field and its rotations will conform with the charts when the distortion weight decreases.

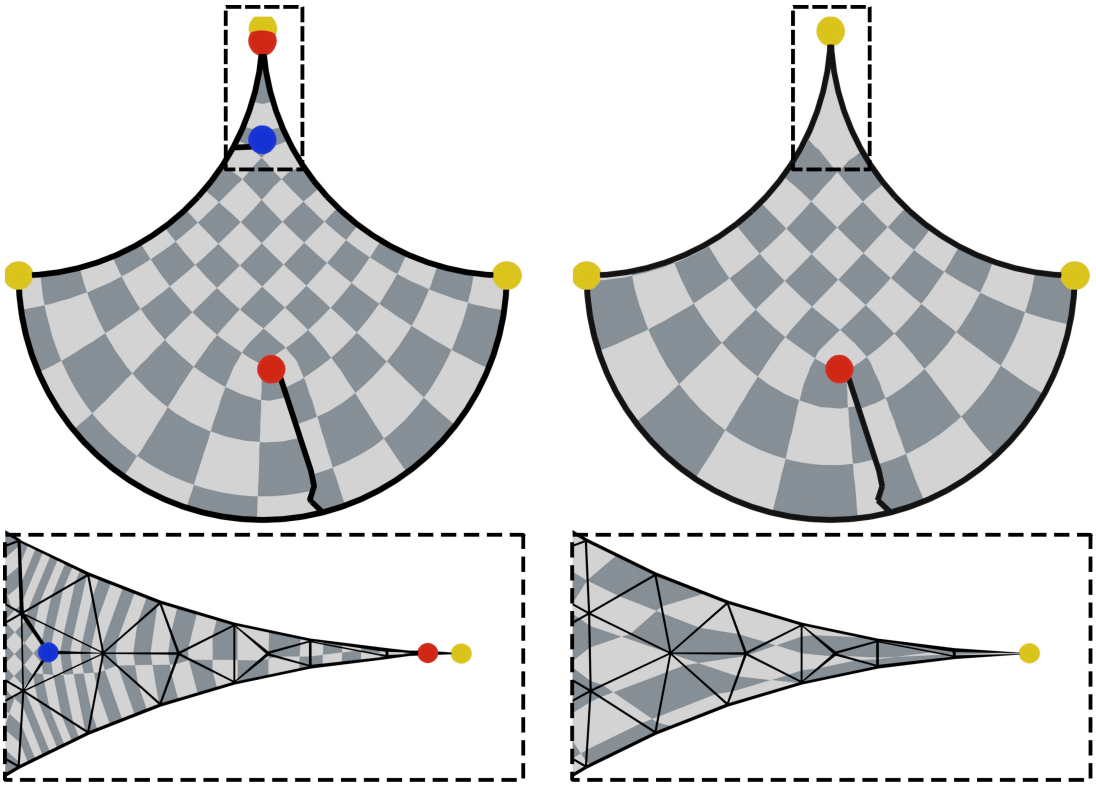
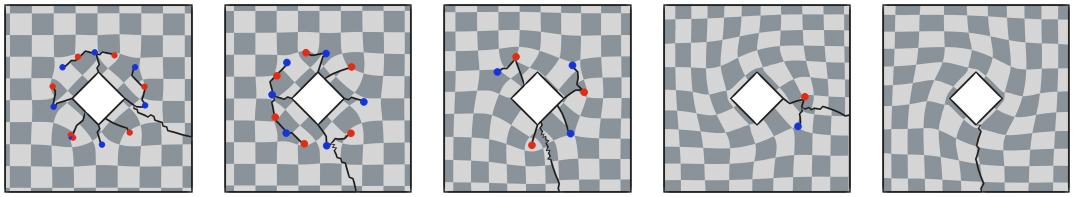


Figure 4.11: Influence of the initialization of the cross field rotations on a sharp corner parametrization. Left: Initialization with zeros rotation. Right: Initialization with modified parallel transport prevents a dipole of singularities to appear and reduces overall distortion. Singularities $\frac{\pi}{2}$ are depicted in red, $-\frac{\pi}{2}$ in blue and boundary corners in yellow.

4.8.4 Parameter Choice

Since our objective function of Equation (4.11) is a sum of several terms, their balance influences the final solution. While every global parametrization is a zero of the energy $\|\mathcal{E}\|_2^2 + \lambda_F \|\mathcal{F}_n\|_2^2$, the cone distribution depends on the value of λ_F. For small λ_F, the chart term $\|\mathcal{E}\|_2^2$ converges first. The optimizer prioritizes the validity of the parametrization over the quantization of singularities to multiple of $2\pi/n$. This results in parametrizations with more cones, and often smaller distortion. On the other hand, for large λ_F, the optimization prioritizes the convergence

of the frame field variables and the quantization of rotations. The final parametrization exhibits fewer singularity cones and a smoother frame field. This phenomenon is illustrated in Figure 4.12, where the same mesh is parametrized for six values of λ_F. As Myles and Zorin [2012], we observe that adding well-placed singularities tend to decrease the distortion. In our experiments in Sec. 4.9, our default value is $\lambda_F = 10$ representing an interesting trade-off. The other weighting term λ_{det} is set to 1.



λ_F	0.01	0.1	1	10	500
Cones	14	12	6	2	0
Scale	1.007	**1.006**	1.011	1.007	**1.006**
Stretch	1.086	**1.085**	1.201	1.405	1.470

Figure 4.12: Influence of the parameter λ_F on the singularity configurations when we optimize for the constraints without distortion. Small values of λ_F produce more singularity cones and minimum stretch. In contrast, large values create smoother frame fields and higher stretch.

4.9 Applications and Evaluation

In this section, we present results obtained with our method on a large selection of models and comparison with previous works.

Database and implementation. Our results are obtained on a database of *274* triangular meshes. This includes *131 models without feature edges* gathered from Myles et al. [2014] and Levi [2021] or created manually for specific test cases (like the model of Figure 4.14). Additionally, we collected *143 CAD models* with correct feature edge flagging from the Mambo CAD dataset[9] as well as models with sharp edges from Myles et al. [2014].

All our experiments were conducted on a Ubuntu 18.04 workstation with a eight-core, 2.6-GHz Intel Core i5. Our implementation was done in python and accelerated with Numba [Lam et al. 2015]. We solve the quadratic programming problems inside the Levenberg-Marquardt algorithm using the open source solver OSQP [Stellato et al. 2020] linked to the Pardiso linear solver [Alappat et al. 2020; Bollhöfer et al. 2019, 2020]. Computation times for the optimization without distortion are displayed in Figure 4.13a.

4.9.1 Seamless Parametrization

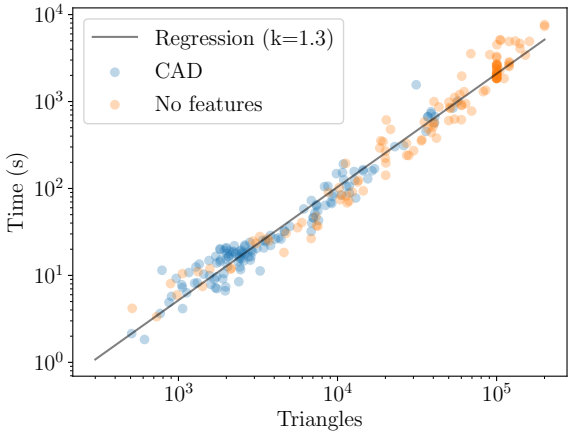
We first setup our algorithm to compute a rotationally seamless parametrization for all of the 274 models. For the 131 models with no features, we output a valid parametrization for 123 of them. Since we need to avoid the trivial zero solution, we initialize our optimization with a smooth frame field (as described in Section 4.8.3). For the 143 models with feature edges, we obtain 132 valid results. We initialize those models with fixed frames on feature edges and zero elsewhere.

[9] `https://gitlab.com/franck.ledoux/mambo`

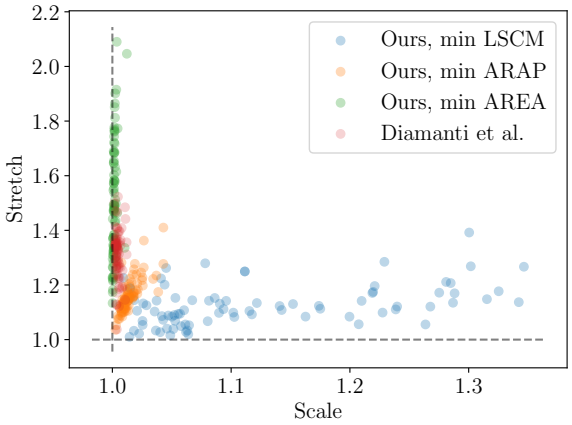
The 19 failure cases observed in the whole database are of two types.

1. Low quality triangles. 17 of the 19 failed models present flat or nearly-flat triangles, with angles as low as 1e-4 radians. This creates numerical instabilities or infinite values in the log-barrier B_η (Equation (4.8.1)). A simple local remeshing to get rid of bad quality elements was sufficient to ensure convergence.

2. Limit Cycles. The two remaining cases are due to topological failure preventing the convergence of the optimization scheme. This is typically created by a frame field with a *limit cycle* whose parametrization is degenerated. Figure 4.14a shows an example of such a configuration. In this case, the algorithm is not able to converge towards a zero of the objective function but tries to reach a global minimum at infinity. Standard parameters of our optimization are not able to escape this situation. However, we can lower the weight on the frame field constraint ($\mathcal{F}_n$) to $\lambda_F = 0.01$, so that the edge constraint ($\mathcal{E}$) dominates the energy (see Section 4.8.4). This way our algorithm converges to a global minimum and introduces new singularities as in Figure 4.14b.



(a) Running time (in seconds) with respect to triangle count for surfaces with (142) and without (132) features.

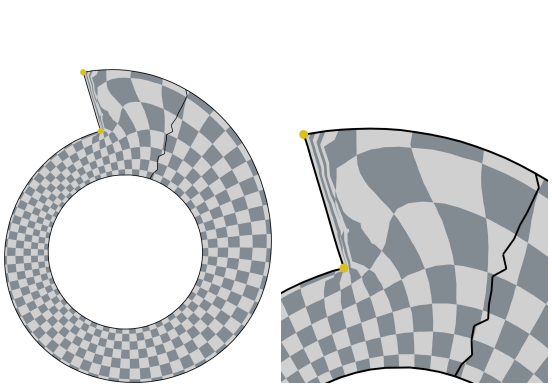


(b) Scatter plot of total stretch vs total scale observed on each models provided by Diamanti et al. [2015]. We consider three optimization scenarios: minimization of $\mathcal{D}_{\text{ARAP}}$, $\mathcal{D}_{\text{LSCM}}$ and $\mathcal{D}_{\text{LSCM}}$.

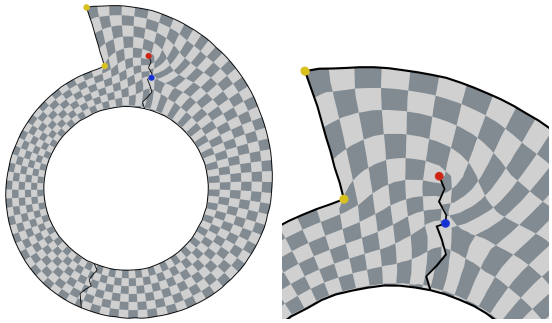
4.9.2 Other Applications

Besides seamless parametrization, we now demonstrate the versatility of our method through a series of other applications.

Cone placement of arbitrary indices. Using Equation ($\mathcal{F}_n$), we can constrain cone singularities to any user-chosen value of $\pm 2\pi/n$ by changing the parameter n. While choosing $n = 4$ leads to rotationally seamless maps used for quad remeshing, higher order can be easily computed by our method with minimal change. On Figure 4.15, we present two models, one with feature and one without, that are parametrized using $n = 3$,6 and 8. Higher values of n naturally result in a greater number of singularity cones, as well as less distorted parametrizations at the expense of more cones and seams.

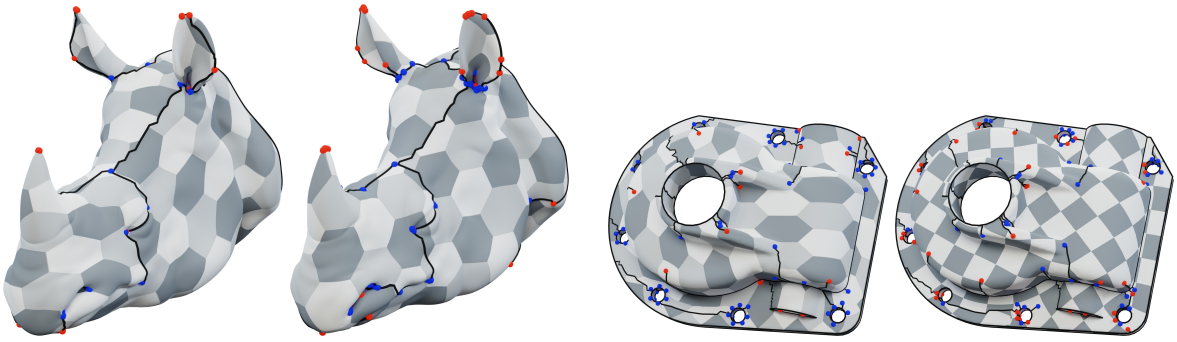


(a) Typical failure case of our method.



(b) Changing the energy weights allows our method to place a dipole of singularities and retrieve a correct parametrization.

Figure 4.14: On this model our method fails to find a valid parametrization ($\lambda_F = 10$) (a). By setting $\lambda_F = 0.01$, we are able to find a valid solution (b).



55 Cones
Scale=1.019
Stretch=1.206
(a) *Rhino*, $n = 3$

138 Cones
Scale=1.004
Stretch=1.087
(b) *Rhino*, $n = 6$

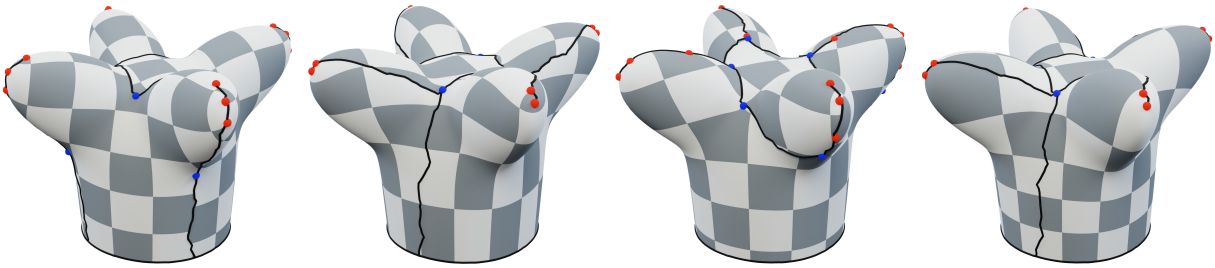
160 Cones
Scale=1.019
Stretch=1.295
(c) *Casting*, $n = 6$

166 Cones
Scale=1.006
Stretch=1.143
(d) *Casting*, $n = 8$

Figure 4.15: Results of our algorithm on two models for different singularity cones quantized to $\pm 2\pi/n$ for $n = 3, 6$ and 8. On the right hand side, parametrizations of the *Casting* model are aligned with its feature edges, on which odd values of n are incompatible due to π not being a valid corner angle. When n is multiple of 3, we represent the parametrizations using a hexagonal pattern, instead of the classical checker pattern.

Fixed topology parametrization. Finally, it is possible to lock the frame field and rotations variables v and ω during optimization in order to compute parametrization with a user prescribed cone distribution and alignment constraints. Figure 4.1b shows a parametrization of Rhino's head based on the principal curvature directions.

This allows us to investigate the benefits of simultaneously optimizing for the cone distribution and the parametrization geometry. In Figure 4.16, we run our algorithm with the *fixed* frame field generated by Viertel and Osting [2018]. We compare the resulting parametrization to our full algorithm using the three distortions of Section 4.8.2. We observe that, by minimizing LSCM, our method outputs a parametrization with less cones that better minimizes the *stretch* distortion at the price of more *scale* distortion. The opposite holds when minimizing the area energy. Again, the ARAP energy yields the most isometric deformation and thus balances both distortions. Interestingly, a smooth frame field is also an excellent proxy to obtain area-preserving maps.



20 Cones Scale = 1.005 Stretch = 1.093	12 Cones Scale = **1.001** Stretch = 1.698	24 Cones Scale = 1.021 Stretch = 1.177	12 Cones Scale = 1.236 Stretch = **1.018**
(a) Fixed singularities	(b) Minimizing $\mathcal{D}_{\text{AREA}}$	(c) Minimizing $\mathcal{D}_{\text{ARAP}}$	(d) Minimizing $\mathcal{D}_{\text{LSCM}}$

Figure 4.16: Comparison between a parametrization with fixed cone singularities from a smooth frame field [Viertel and Osting 2018] and the one obtained by optimizing cones placement. Minimizing distortion leads to different cone placements. (b) Minimizing $\mathcal{D}_{\text{AREA}}$ effectively reduces the scale distortion (c) Minimizing the ARAP energy results in a tradeoff with a different topology. (d) Minimizing the LSCM energy outputs a parametrization that is almost conformal.

Free boundary parametrization. So far, we focused on computing a parametrization with a cone metric where cones are constrained to have $2\pi/n$ curvature, with $n > 1$. By setting $n = 1$ in Equation $(\mathcal{F}_n)$, cones are therefore forced to take values of either 2π, which implies that at least one chart will be flattened or flipped, thus contradicting the injectivity condition of Equation $(\mathcal{I}_1)$, or -2π, which implies a "double covering" which is impossible in our setting (see Theorem 4.3). Therefore, no singularity can appear in a parametrization that satisfies Equation $(\mathcal{F}_n)$ for $n = 1$. Besides, by omitting boundary constraints of Equation $(\mathcal{B})$ in the optimization, our algorithm is able to produce free boundary parametrization with minimal distortion for disk-shaped models. Figure 4.17 presents four of such parametrizations.

4.9.3 Comparison With Other Methods

We now provide comparisons with previous works computing rotationally seamless maps for $n = 4$ by considering our two distortion metrics *scale* and *stretch*. For a more precise analysis than just an integrated value, we plot distortion histograms in log scale of the triangle count. When quad meshes are provided, they are obtained by computing an integer grid map using the method of Bommes et al. [2013a] and by then extracting quads following Ebke et al. [2013]. Most importantly, they have the same singularities (positions and indices) as the input parametrizations.

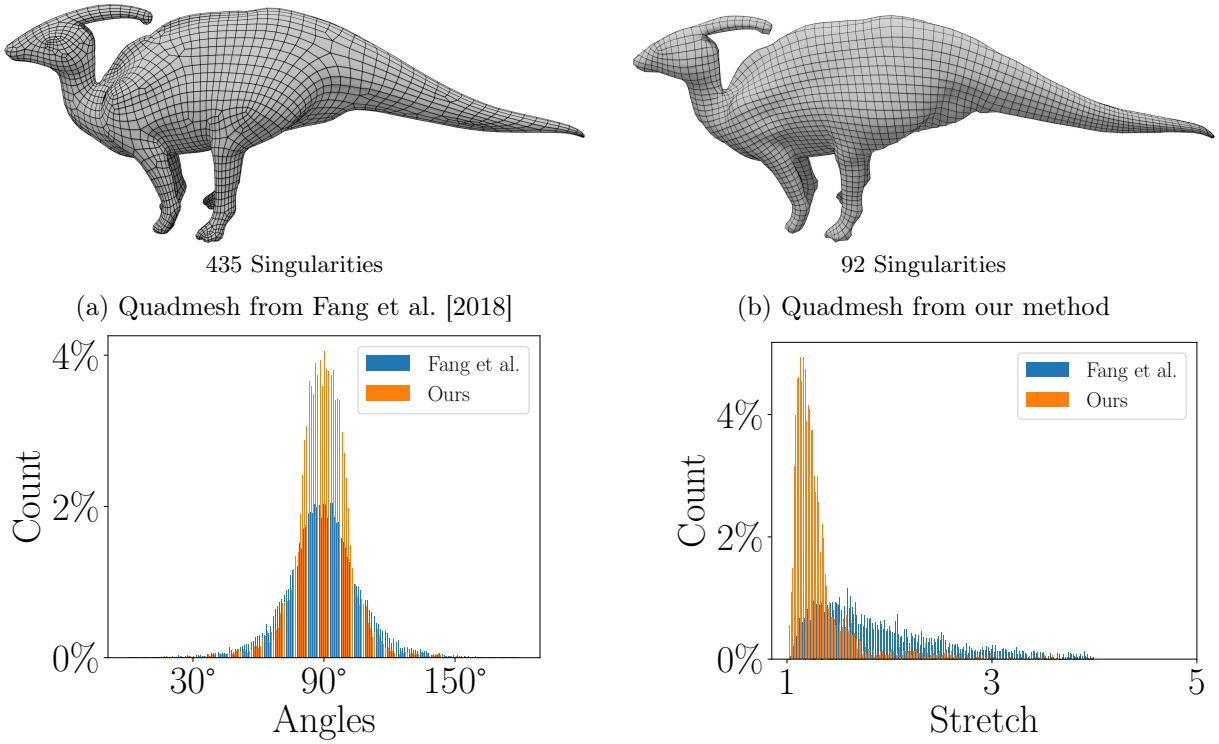
Comparison with Levi [2021] Levi directly computes a seamless parametrization from a triangle mesh using a greedy algorithm. However, this method seems very dependent on its initialization: for the *Dilo* model in Figure 4.20, the singularities appear on a single side of the mesh and seem unrelated to the geometry. The parametrizations provided by the author minimize the maximum stretch over all triangles. This can be observed in Figure 4.19 where the stretch is bounded. In comparison, our method yields a higher maximal value but a smaller stretch for a majority of triangles (see histograms), as well as a reduced number of cone singularities – 16 against 217. Our optimization does not attempt to reduce triangle scaling in this case, thus our scale distortion is not as competitive. In Figure 4.21, we observe that our obtained quads are overall slightly less distorted for a smaller number of singularities (73 against 105).

Figure 4.17: Free boundary parametrization using our algorithm on disk-shaped models (top), or models with user-prescribed seams (bottom).

Comparison with Diamanti et al. [2015] The integrable Polyvector field method takes an initial frame field and optimizes it to guarantee its integrability. However, this process is prone to introduce new singularities. In Figure 4.13b, we compare our method on the 62 meshes provided by the authors. The integrable Polyvector field method performs very well for scale minimization. However, we are able to outperform them while placing 15% less cones on average.

Comparison with Myles and Zorin [2012] Myles and Zorin propose a greedy algorithm for cone placement along with a post-processing procedure to obtain $n\pi/2$ singularity cones. Cones tend to be placed in high curvature regions which can be suboptimal. In Figures 4.20, our method is shown to match their scale distortion and outperform their stretch distortion on the *Dilo* model.

Comparison with Fang et al. [2018] Finally, we compare our results to a method producing quad-meshes without the quantization step. Fang et al. [2018] use periodic functions to integrate



(a) Quadmesh from Fang et al. [2018]

(b) Quadmesh from our method

(c) Angles histograms

(d) Stretch distortion histograms

Figure 4.18: Quad remeshing comparison with Fang et al. [2018] on the *Dilo* model. Our parametrization was initialized using principal curvature direction and optimized without distortion. Our quads have less stretching and have angles closer to 90°.

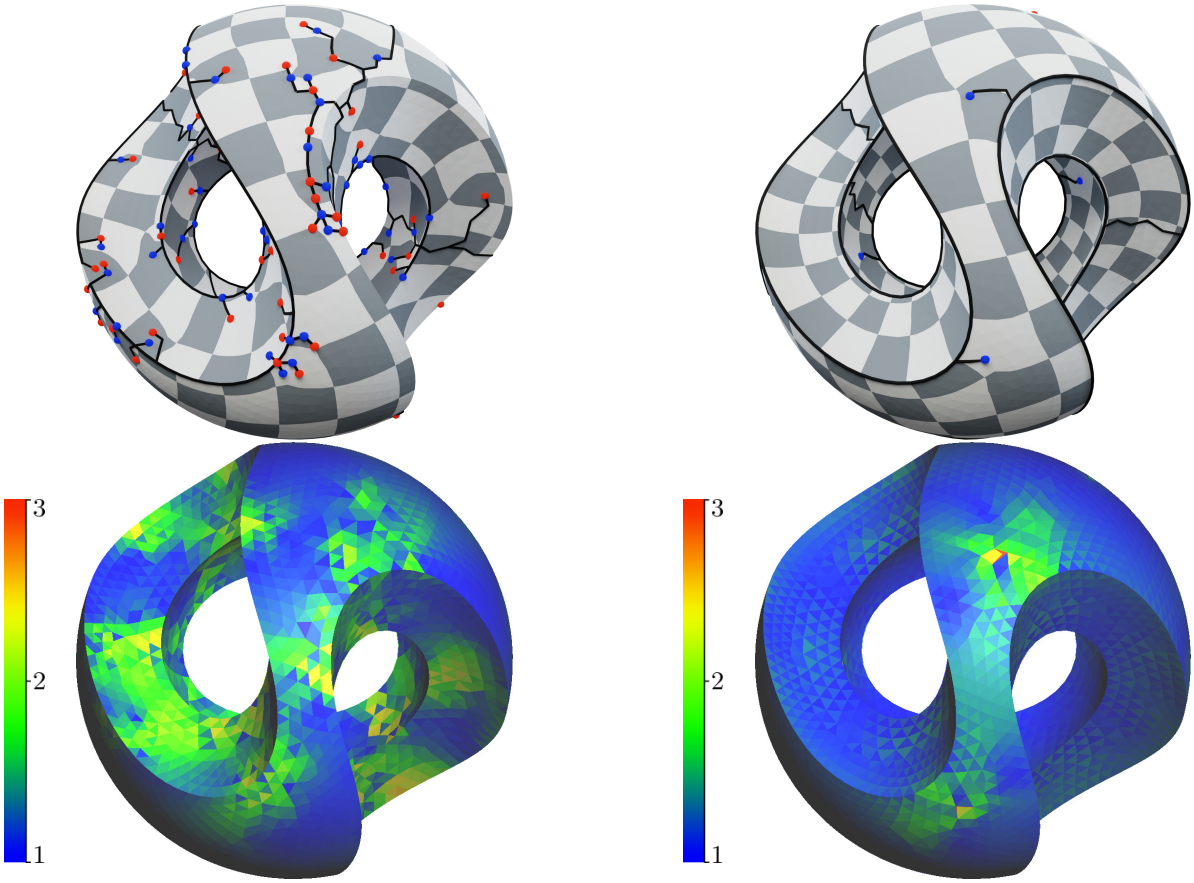
a frame field directly into an integer seamless parametrization. This type of methods generally output a high quality mesh on most of the input surface. However, some localized regions need to be repaired which often introduce new singularities.

Figure 4.18 shows the repartition of quad angles and stretch for our respective quad meshes of the *Dilo* model. Our quads are more orthogonal and suffer less from stretching, while our meshes present fewer singular vertices. The same conclusion is drawn from Figure 4.21 where ours quads are closer to perfect squares for less singularities overall (168 against 73), although Fang et al. [2018] better capture curvature alignment around the leg of the model.

4.10 Conclusion and Perspectives

By discretizing Cartan's structure equations, we were able to construct a general algorithm for global parametrization where cones and seams do not need to be known in advance, but can instead be computed along with the geometry. This all-in-one optimization of the topology and the geometry enables us to find cone points whose positions minimize any given distortion energy. Unlike most previous methods our solver is not greedy and does not rely on integer variables.

In the perspective of quad meshing, the main limitation of our work is that it does not handle integer coordinates of cone points. We still rely on a post-processing quantization step which impacts the overall distortion distribution. While the cone placement is optimal for the seamless parametrization, there is no guarantee of optimality for the final quadmesh.



217 Cones
Scale = 1.025
Stretch = 1.422

(a) *Sculpt* model, [Levi 2021]

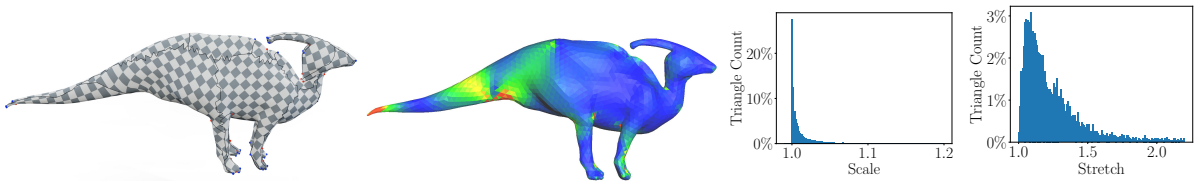
16 Cones
Scale = 1.146
Stretch = 1.185

(b) *Sculpt* model, ours (min LSCM)

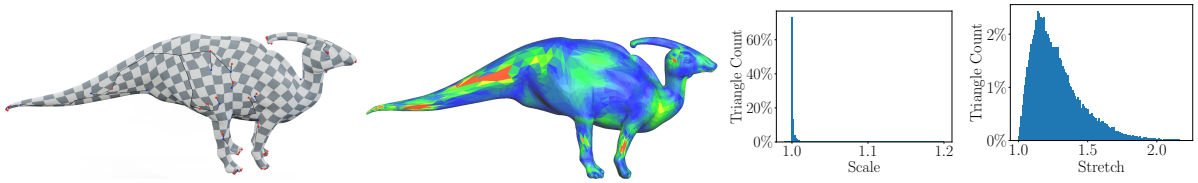
Figure 4.19: Comparison with Levi [2021] on the sculpt model. Both methods minimize the stretch distortion. As shown by the heat map and the histogram, our method achieves comparable triangle stretch with ten times less cone singularities.

A second limitation is that our current implementation is slower than previous methods because of the a higher number of variables: $2|F| + 4|E|$ for vertex charts coordinates, $|E|$ for the parallel transport and $2|V|$ for the cross field.

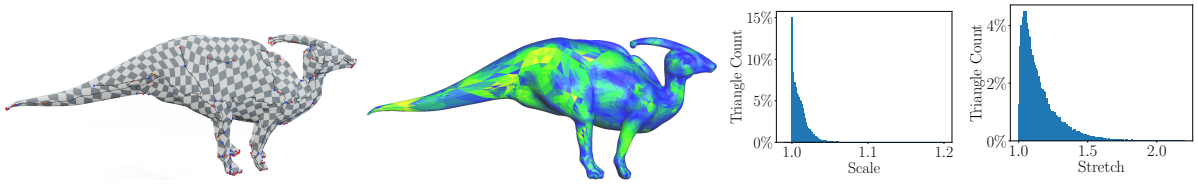
Finally, since Cartan's method of moving frame is not limited to two-dimensional geometry, the structure equations can be written for higher dimension meshes. This opens the door to future works for computing valid seamless parametrization for tetrahedral meshes in the context of hexahedral remeshing.



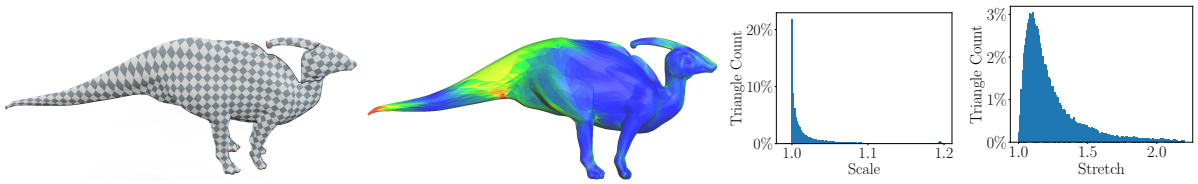
(a) [Myles and Zorin 2012], 82 cones



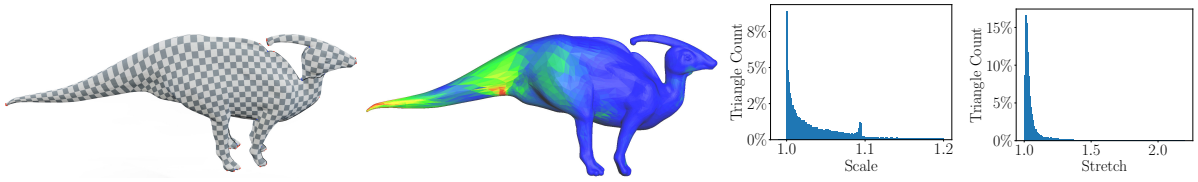
(b) [Diamanti et al. 2015], 140 cones



(c) [Levi 2021], 204 cones



(d) Ours (no distortion), 78 cones



(e) Ours (min LSCM), 68 cones

Figure 4.20: Comparison of three baseline methods with our results on the dilo model. Top row: texture mapping with highlighted seams and singularities . Second row: stretch distortion distribution over the model from 1 (blue) to 3 or higher (red). Third row: histogram of stretch distortion on triangles. Our method is best at minimizing the stretch distortion even compared to the conformal algorithm of Myles and Zorin [2012].

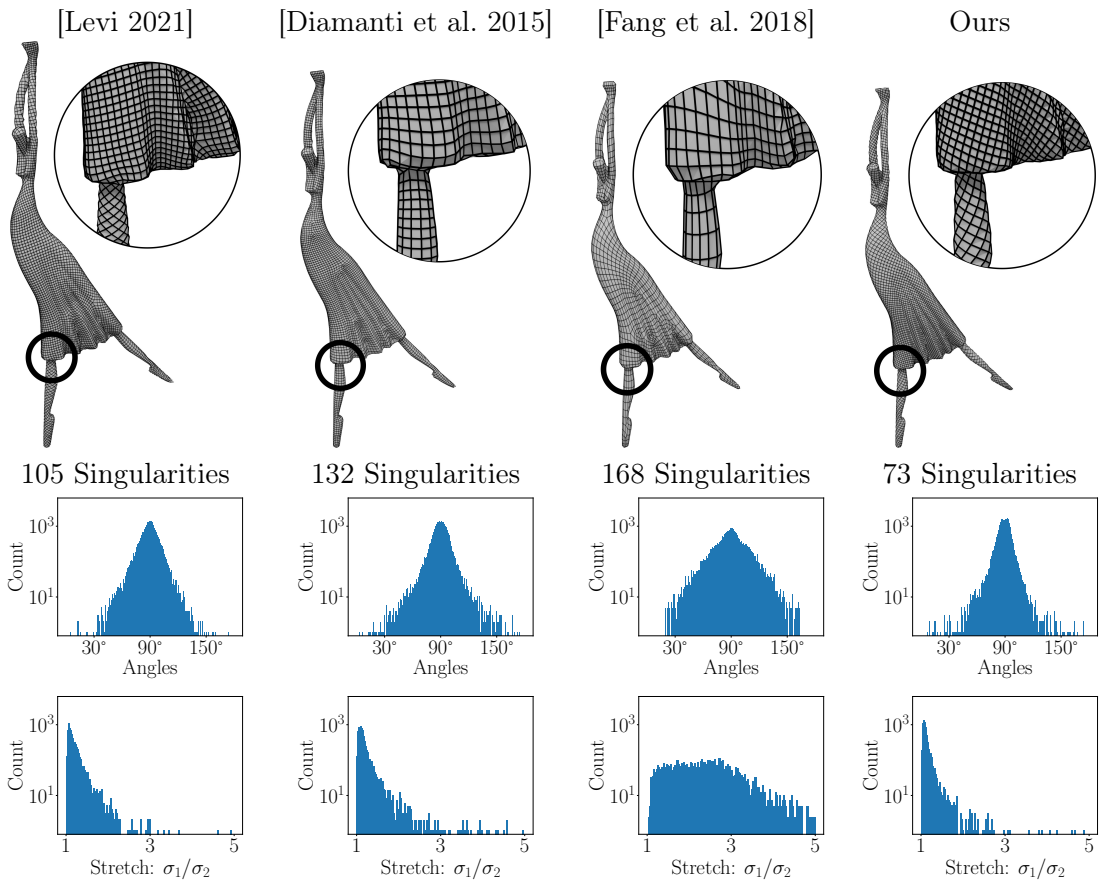


Figure 4.21: Comparison of quadmeshes obtained by three concurrent algorithms with ours on the dancer model. Top row: quad meshes. Second row: angle distribution over the model. Third row: histograms of stretch distortion over elements. Our method yields minimal stretch and most orthogonal quads.

5

Alternative Discretization for the Method of Moving Frames

Contents

5.1 Face-based Discretization 120
5.1.1 Discrete Frame Field Condition 120
5.1.2 First Structure Equation 121
5.1.3 Local injectivity 122
5.1.4 Reconstruction 122
5.1.5 Boundary and feature edge constraints 124
5.2 Numerical Optimization 124
5.2.1 Non-Linear Least Squares 124
5.2.2 Initialization 125
5.2.3 Distortion Energy 125
5.2.4 Distortion Minimization Strategies 126
5.3 Results and Comparison 128
5.3.1 Database and Implementation 129
5.3.2 Comparison with a Vertex-based Implementation 129
5.3.3 Distortion Strategies 129
5.4 Conclusion and Perspectives 131

In the previous chapter, we demonstrated how Cartan's method of moving frames could be used to design an algorithm to simultaneously optimize the singularities' positions and the distortion of a seamless parametrization. To this end, we derived a discretization of the Cartan's structure equations as relationships between adjacent vertex-based charts. Yet the choice of such a discretization can be seen as unusual for a seamless parametrization since the output result is naturally expressed in a per-triangle manner. In Section 4.4.1, we established the two benefits of this approach. Firstly, we showed that a solution to our problem was a valid seamless parametrization with the guarantee of local injectivity, since double-coverings cannot appear. Secondly, discretizing on vertices allow singularities to be freely placed inside faces, enabling parametrization of very coarse meshes, as well as more control on their final positions.

Unfortunately, these benefits come at the cost of a more complex formulation and an optimization problem involving more variables. In this chapter, we investigate the simpler face-based

discretization of the structure equations and derive a variant of the moving frame parametrization algorithm. While we cannot expect the same theoretical guarantees about injectivity, we demonstrate a significant improvement in computation time in practice, all while retaining the same quality of results. Moreover, working directly with faces allows other distortion energies to be defined, as well as other minimization strategies to be utilized.

This chapter is organized as the second part of the last. We begin by deriving the face-based discrete structure equations using discrete exterior calculus [Hirani 2003] in Section 5.1. Then, Section 5.2 describes the setup of our optimization problem. Finally, Section 5.3 will report the results of various experiments made to compare the two versions of our algorithm. Based on them, we will conclude that contrary to our intuition, having singularities not restricted at vertices was not necessary to achieve convergence and good results, as this face-based version presents comparable and even sometimes better results in terms of final distortion.

5.1 Face-based Discretization

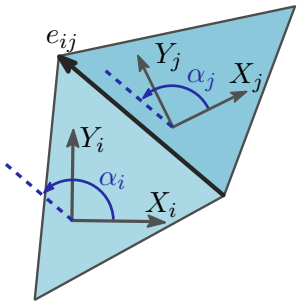
Let us start by recalling the theoretical problem established in Section 4.2, Equation (4.6). Given a smooth 2D manifold $\mathcal{M}$, The goal is to find local frames z, rotations ω and local deformations σ solution of:

$$\begin{array}{lll} \min & \mathcal{D}(\sigma) & \\ z : \mathcal{M} \to \mathbb{U}/\Gamma & & \mathrm{d}z - \imath\omega z = 0 \\ \omega : \mathcal{M} \to \mathbb{R} & \text{s.t.} & \mathrm{d}\sigma + \imath\omega \wedge \sigma = 0 \\ \sigma : T\mathcal{M} \to \mathbb{C} & & \star\mathrm{Im}(\bar{\sigma} \wedge \sigma) > 0 \end{array} \tag{5.1}$$

where $\mathcal{D}$ is a smooth distortion energy.

5.1.1 Discrete Frame Field Condition

Parallel transport. As before, let $M = (V, E, T)$ be a triangle mesh, consisting of vertices, edges and triangles. We recall the face-based parallel transport computation exposed in Section 2.2.2 of Chapter 2. Each face $t_i \in T$ is equipped with an arbitrary local basis X_i, Y_i spanning its tangent space. Following Crane et al. [2010], we can define the parallel transport $\rho : E \to \mathbb{R}$ between two adjacent triangles t_i and t_j as the angle difference between representations of their common edge e_{ij} in their respective bases:



$$\rho_{ij} := \alpha_j - \alpha_i$$

where α_i (resp. α_j) is the angle between e_i^j and X_i (resp. e_j^i and X_j). The change of basis is then again $r_{ij} = \exp(\imath\rho_{ij})$. Just like its vertex-based counterpart, this parallel transport defines an integrated Gaussian curvature $K \in (-\pi, \pi]$ defined around vertices as :

$$\exp(\imath K_v) = \prod_{e^{ij} \in \mathrm{ring}(v)} r_{ij}$$

where the product is taken over all adjacent edges of v.

Frame field condition. The discretization of the frame field condition $dz = \imath\omega z$ is similar to the one described in Section 4.4. We assign to each triangle a frame z represented by a unit complex number. For two adjacent triangles t_i and t_j, the discrete exterior calculus [Hirani 2003] yields a similar equation as before (Equation (4.9)):

$$(1 + \imath\omega_{ij}/2)z_j = (1 - \imath\omega_{ij}/2)r_{ij}z_i =$$

where adjacent frames are related by the rotation $cay(\omega_{ij})$, up to the corresponding parallel transport r_{ij}.

Since we adopt the same frame field representation as the vertex-based version, namely the n directions of a rotationally symmetric frame are represented by the n-th roots of a unit complex $v \in \mathbb{U}$, we can raise this equation to the power n to retrieve the same constraint:

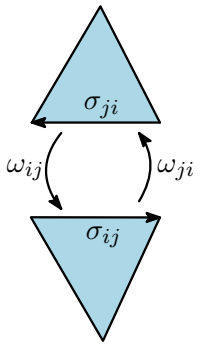
$$\left| \quad (1 + \imath\omega_{ij}/2)^n v_j \quad = (1 - \imath\omega_{ij}/2)^n r_{ij}^n v_i \right. \qquad (\mathcal{F}_n)$$

5.1.2 First Structure Equation

Now, we turn our attention to the *first structure equation* of Equation (4.3). Following the same path as in the vertex-based version, we define a "lens" complex by gluing together opposite half-edges of adjacent triangles. The local deformation σ is defined per triangle, and σ_{ij} (resp. σ_{ji}) will denote the deformed edge ij in the respective bases of triangles t_i and t_j. By applying the parallel transport r_{ij} defined earlier, we can express the integral exterior derivative $d\sigma$ as:

$$(d\sigma)_{ij} = r_{ij}\sigma_{ji} - \sigma_{ij}.$$

The wedge product can be interpreted, just as the vertex-based version, as the wedge product on a virtual quadrangle formed by the two opposite half-edges, where $\omega = 0$, and two virtual edges linking their extremities, where $\sigma = 0$:



$$(\omega \wedge \sigma)_{ij} = \imath\omega_{ij}(\sigma_{ij} + r_{ij}\sigma_{ji})/2.$$

By combining those two expressions, we arrive at the discretized version of the first structure equation:

$$(1 + \imath\omega_{ij}/2)\sigma_{ij} = (1 - \imath\omega_{ij}/2)r_{ij}\sigma_{ji} \qquad (5.2)$$

where two opposite half-edges are related by the inverse Cayley transform $cay^{-1}(\omega_{ij})$ and the inverse parallel transport. Moreover, since $\omega = 0$ inside triangles, the integral exterior derivative around a triangle yields:

$$d\sigma = 0 = \sigma_{ij} + \sigma_{jk} + \sigma_{ki}$$

which simply means that the image of the triangle t_{ijk} by σ remains a valid triangle. Thanks to this fact, it is possible to represent σ_i as a linear transformation of the original triangle. In practice, we define a matrix $J \in \mathbb{R}^{2\times 2}$ per triangle, and we consider that edges in parameter space are images, by this matrix, of the original edges of the mesh expressed in the local basis. In other words, if e_i^j and e_j^i are the coordinates of edges e_{ij} in bases X_i, Y_i and X_j, Y_j then $\sigma_{ij} = J_i e_i^j$ and $\sigma_{ji} = J_j e_j^i$. Using an abuse of notations, we identify the complex multiplication by $a + \imath b$ with the matrix multiplication by $\begin{pmatrix} a & b \\ -b & a \end{pmatrix}$. This allows writing Equation (5.2) as follows:

$$\Big| \; (1 - \imath\omega_{ij}/2) r_{ij} J_j e_j^i = (1 + \imath\omega_{ij}/2) J_i e_i^j \qquad (\mathcal{J})$$

Therefore, both versions of an edge in corresponding triangles should match in parameter space up to the rotation $cay^{-1}(\omega_{ij})$ and the inverse parallel transport.

5.1.3 Local injectivity

In order to ensure local injectivity, we require that areas of triangles in the parametrization stay positive. Given our formulation with Jacobian matrices, this boils down to a simple positivity constraints on matrices' determinants:

$$| \det(J) > 0 \qquad (\mathcal{I})$$

5.1.4 Reconstruction

Now that we detailed the expression of our system of equations in this face-based discretization, we focus on the parametrization reconstruction. Given a set of Jacobian matrices J_t that satisfy Equations $(\mathcal{J})$, $(\mathcal{F}_n)$ and $(\mathcal{I})$, we will show that we can extract a valid seamless parametrization. We can indeed use a very similar algorithm as the vertex-based reconstruction described in Section 4.6 with the difference that triangles are readily available as single charts and singularity cones can only appear at vertices. The reconstruction scheme is therefore simpler and can be done in four steps:

1. Reconstruct triangles independently by applying their matrix J_t to their initial coordinates;
2. Recover a basis z per triangle from the power vector v by finding the smallest symmetry rotation along a tree;
3. Rotate each triangle with relation to their respective bases;
4. Recursively stitch adjacent triangles along the tree.

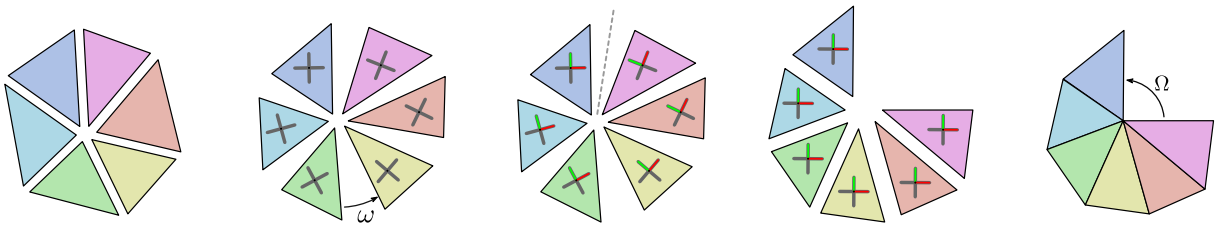


Figure 5.1: Reconstruction of a parametrization from triangle charts. Original triangles of the mesh (a) are deformed independently by their respective Jacobian matrix (b). These triangles correspond to each other by the cross field rotation $cay(\omega)$. From the cross field, local frames can be extracted by finding the best matching along a tree (c). This creates seams when needed. By aligning all the local frames (d), triangles can be stitched back together and produce a parametrization which defects are exactly prescribed by the cross field (e).

This process is summarized in Figure 5.1. Moreover, the following theorem holds:

Theorem 5.1. *If a set of Jacobian matrices per triangle J, rotations ω and frame field v satisfy Equations $(\mathcal{J})$, $(\mathcal{F}_n)$ and $(\mathcal{I})$, then we can recover a parametrization with quantized cones whose transition functions are rotation equal to the frame field discontinuities. Local injectivity is guaranteed up to the possible appearance of double-coverings.*

Proof. The proof is very similar to the one of Theorem 4.3. We show that parametrization cuts appear at frame field discontinuities, and that the parametrization's defect matches the jump of the frame field.

To this end, consider two adjacent triangles t_i and t_j with common edge e. According to Equation $(\mathcal{F}_n)$, their respective frames z_i and z_j are related by the following equation:

$$z_j = \exp\left(2\imath k_e \pi/n\right) cay(\omega_{ij}) r_{ij} z_i$$

where $k_e \in \mathbb{N}$ is the jump index of the frame field at edge e. Let $a = z_i \sigma_{ij} = z_i J_i e_i$ and $b = z_j \sigma_{ji} = z_j J_j e_j$ be the images of edge e by the two local deformations in aligned bases. Using Equation $(\mathcal{J})$, we can compute:

$$\begin{aligned} b &= z_j J_j e_j \\ &= \exp\left(2\imath\pi k_e/n\right) cay(\omega_{ij}) r_{ij} z_i \left(cay^{-1}(\omega_{ij}) r_{ji} J_i e_i^j\right) \\ &= \exp\left(2\imath\pi k_e/n\right) z_i J_i e_i \\ &= \exp\left(2\imath\pi k_e/n\right) a. \end{aligned}$$

Hence the two versions of the edge match up to the frame field discontinuity $\exp\left(2\imath\pi k_e/n\right)$. The transition functions of the parametrization match the rotation of the frame field, which makes the parametrization seamless for $n = 4$. □

Contrary to the vertex-based version of the proof, it is impossible in this case to prevent double coverings from happening. Figure 5.2 presents a test case where such a configuration appears in our algorithm.

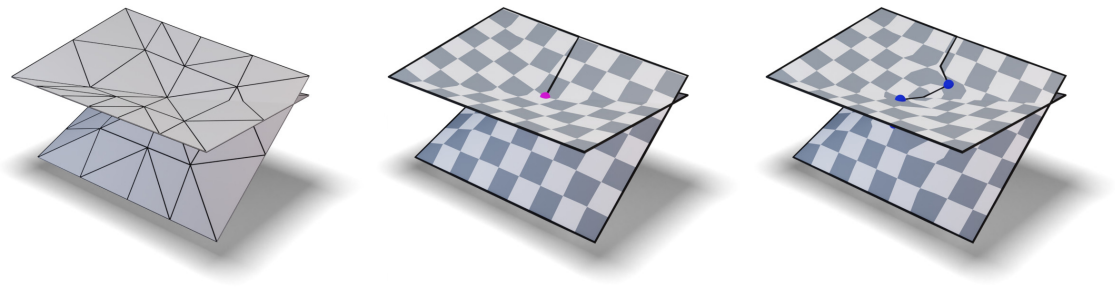


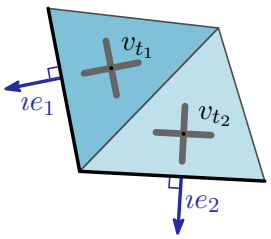
Figure 5.2: Parametrization obtained by the two versions of our algorithm on a mesh with a central vertex of angle defect close to -2π (left). The face-based algorithm converges to a solution with a single -2π singularity cone in the middle, thus creating a double-covering (middle). On the other hand, this behavior is impossible in the vertex-based algorithm, which instead creates four $-\pi/2$ singularity cones in the triangle ring of the central vertex (right).

5.1.5 Boundary and feature edge constraints

It is now time to discuss the additional constraints required on the parametrization to be *adapted* to the mesh boundary and feature edges. Unlike the vertex-based version in Section 4.7, we do not need the target angle of feature corners.

Let $t \in T$ be a triangle adjacent to a feature edge e. Let α_e be the angle between edge e and basis vector X_t. Let J_t and v_t be the Jacobian matrix and frame variables at triangle t.

For the parametrization to be adapted to the feature edge e, we require that the edge coordinates in parameter space remain orthogonal to the feature edge normal $\imath e$, and that one of the frame's direction should be aligned with the edge. In other words, the following two conditions should hold:



$$\left| \begin{array}{l} v_t = \exp(n\imath\alpha_e), \\ \langle J_t e, \imath e \rangle = 0 \end{array} \right. \tag{$\mathcal{B}$}$$

Note that in a face-based setting, it is necessary for triangles to be adjacent to at most one feature or boundary edge, since the frame can be aligned to only one edge (with the exception of the particular case of orthogonal edges). We therefore split triangles adjacent to two or more feature edges using an additional central vertex. That way, we guarantee a sufficient number of degrees of freedom in the area for the algorithm to converge.

5.2 Numerical Optimization

We now turn our attention towards the numerical optimization of the face-based discretization of Problem (5.1).

5.2.1 Non-Linear Least Squares

Given Jacobian matrices J per triangles, frame representations v and rotation angles ω, the constrained optimization problem reads:

$$\begin{array}{ll} \min\limits_{\omega,v,J} & \mathcal{D}(J) \\ \text{s.t.} & (\mathcal{J}), (\mathcal{F}_n), (\mathcal{B}), (\mathcal{I}) \end{array} \tag{5.3}$$

where $\mathcal{D}$ is any smooth distortion energy on triangles. We again relax this problem to a non-linear least-square form by reformulating the constraints as an energy function:

$$R(\omega, v, J) = \|\mathcal{J}\|^2 + \lambda_F \|\mathcal{F}_n\|^2 + \lambda_{\text{det}} \|B_\eta(\det J)\|^2$$

leading to a non-linear least-square optimization with linear constraints:

$$\min_{\omega,v,J} \quad \varepsilon\mathcal{D}(J) + R(\omega, v, J) \qquad \text{s.t. } (\mathcal{B}). \tag{5.4}$$

Problem (5.4) is optimized using the same Levenberg-Marquardt implementation [Marumo et al. 2020] and the same minimization scheme for ε we used in Chapter 4 (Section 4.8).

5.2.2 Initialization

Variables of the face-based version consist of one Jacobian matrix J per triangle, one frame v per triangle and one rotation angle ω per (dual) interior edge.

Jacobian matrices. Matrices per triangle are simply initialized as the identity matrix. Unlike the vertex-based chart, no initial flattening of the angle defect or additional computation is required. In addition to the final algorithm being simpler, this also provides an advantage in distortion minimization, as the exact piecewise linear distortion is available for optimization, unlike the vertex-based version where this initial flattening was unavoidable. Moreover, this initialization is already a global minimum of the distortion which was not the case a priori for the vertex-based. This allows us to define a very effective distortion metric evaluating our distance to this solution (Equation (5.5)).

Rotations. For surfaces without boundary or feature edges, rotations are simply initialized at zero. Otherwise, we use the same parallel transport correction scheme as in the vertex-based version (Section 4.8.3), following the work of Desobry et al. [2022].

Frame field. The same frame field initializations are available to this version, in particular: starting from zero everywhere except near feature edges, computing a smooth frame field as in [Viertel and Osting 2018], computing principal curvature directions or initializing at random. By default, when no feature or boundary edges are present, we choose to initialize using a smooth frame field for ease of optimization. For models with features, we initialize at zero and rely on the frames propagating from the feature alignment for convergence.

5.2.3 Distortion Energy

Distortion energies defined for vertex charts in Section 4.8.2 can still be used in this context as functions of the triangles' Jacobian matrices. We therefore consider $\mathcal{D}_{\text{ARAP}}$, $\mathcal{D}_{\text{LSCM}}$ and $\mathcal{D}_{\text{AREA}}$ just as before:

$$\mathcal{D}_{\text{ARAP}}(J) = \sum_{t \in T} \|J_t^T J_t - I_2\|^2$$

$$\mathcal{D}_{\text{LSCM}}(J) = \sum_{t \in T} \|J_t - \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} J_t\|^2$$

$$\mathcal{D}_{\text{AREA}}(J) = \sum_{t \in T} \|\det(J_t) - 1\|^2 + \varepsilon \mathcal{D}_{\text{LSCM}}(J)$$

We recall that $\mathcal{D}_{\text{LSCM}}$ minimizes shearing of the elements in the parametrization, while $\mathcal{D}_{\text{AREA}}$ minimizes their scaling. $\mathcal{D}_{\text{ARAP}}$ is again a good tradeoff between the two, as it minimizes variations in edge lengths.

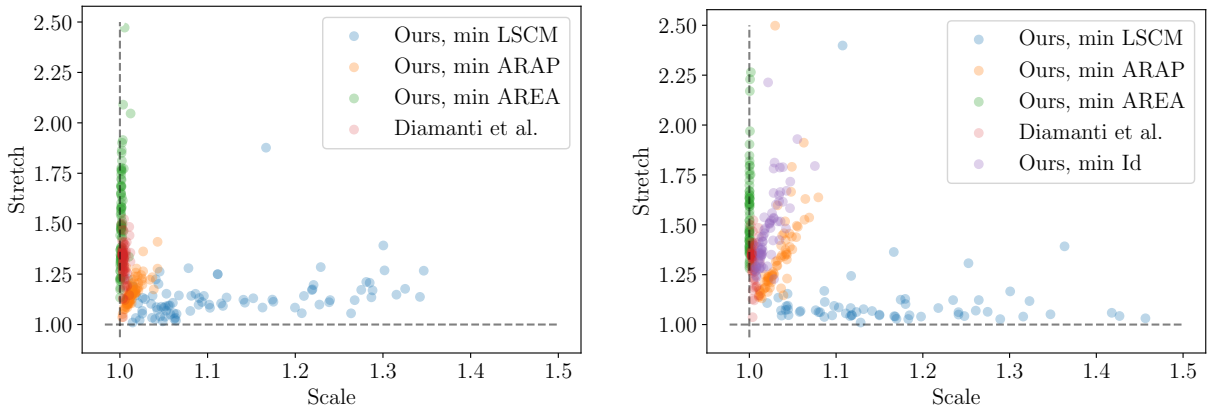
Finally, motivated by the observation that the identity matrices provide a perfectly isometric parametrization, we define the *Identity distortion* where the elements Jacobians should match the identity matrix as much as possible:

$$\mathcal{D}_{\text{Id}}(J) = \sum_{t \in F} \|J_t - I_2\|^2 \tag{5.5}$$

Note that this distortion metric was not relevant in a vertex-based context, since the initial flattening of the vertex rings due to parallel transport meant that the identity was not necessarily an isometry. Here, we have a more direct and total control over the parametrization's distortion.

Results obtained using those four distortions are presented in Figure 5.3 and compared against the vertex-based results on the dataset of Diamanti et al. [2015]. Minimizing the distortion functions $\mathcal{D}_{\text{ARAP}}$, $\mathcal{D}_{\text{LSCM}}$ and $\mathcal{D}_{\text{AREA}}$ in this context lead as expected to a very similar result as before. However, the face-based version is able to go further into the scale/stretch tradeoff with points with even larger stretch (resp. scale) and smaller scale (resp. stretch). This is corroborated by results from Figure 5.6 on another dataset, and probably due to the absence of an initial flattening of charts due to parallel transport.

Finally, as far as the identity distortion is concerned, results are very similar to the ARAP distortion, finding good tradeoffs between scale and stretch and thus focusing on isometric parametrization.



(a) Vertex-based

(b) Face-based

Figure 5.3: Scatter plot of total stretch vs total scale observed on each models provided by Diamanti et al. [2015]. Left: results obtained using $\mathcal{D}_{\text{ARAP}}$, $\mathcal{D}_{\text{LSCM}}$ and $\mathcal{D}_{\text{AREA}}$ with the vertex-based algorithm (reproduction of Figure 4.13b with different axes). Right : results of the face-based version using $\mathcal{D}_{\text{ARAP}}$, $\mathcal{D}_{\text{LSCM}}$,$\mathcal{D}_{\text{AREA}}$ and $\mathcal{D}_{\text{Id}}$. Minimization of distortion behaves mostly in the same way in both cases. The $\mathcal{D}_{\text{Id}}$ has results equivalent to $\mathcal{D}_{\text{ARAP}}$.

5.2.4 Distortion Minimization Strategies

Due to the more direct control we have on triangles' transformations, different strategies are possible in a face-based context. Along the classical least-square minimization of a distortion function, we describe in this section two other methods, namely a change of metrics in the optimizer and delimitation of the feasible region using linear inequalities.

Energy minimization.

Just as the vertex-based method, distortion minimization can be achieved by adding the distortion energy as an additional term in the non-linear least-square optimization. Since the new minima of the energy are no feasible solutions in general, we solve Problem (5.4) with decreasing values of ε, and then one last time without any distortion. The schedule for ε is unchanged, starting from $\varepsilon = 10^2$ and decreasing by a factor 10 until it reaches 10^{-4}.

Change of metric.

In the Levenberg-Marquardt algorithm, an iteration consists in a linear solving involving the previous step and the function's Jacobian. More formally, in order to minimize $||\mathcal{E}(x)||^2$, if x_n is the vector of variables at the current iteration n, $\mathcal{E}_n$ the value of the function at x_n and $\nabla\mathcal{E}_n$ the value of the function's Jacobian, the variables x_{n+1} are found by solving the following system:

$$x_{n+1} = \underset{x}{\operatorname{argmin}} ||\mathcal{E}_n + \nabla\mathcal{E}_n(x - x_n)||^2 + \lambda_n||x - x_n||^2. \tag{5.6}$$

with some sequence of parameters λ_n depending on the implementation [Marumo et al. 2020]. Here, the norm considered is the L^2 norm by default, but it is possible to replace $||x - x_n||^2$ by $(x-x_n)^T\Delta(x-x_n)$ as long as Δ is a symmetric positive definite matrix. Given a Jacobian matrix $J = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, our four distortions can actually be translated in such a 4×4 matrix Δ applied to the parameter vector (a, b, c, d) in order to bias the direction of descent of the optimizer. This method of penalizing distortion has the advantage of only needing a single optimization, contrary to the least-square minimization that utilizes several of them with decreasing values of the weighting parameter ε. This allows for an obvious reduction of the total computation time.

Conformal distortion. The matrix J is conformal if it is the matrix representation of a complex number, meaning that $a = d$ and $b = -c$. To achieve this effect, we change the squared norm to $a^2+b^2+c^2+d^2+\delta(a-d)^2+\delta(b+c)^2$. This is expressed by the quadratic form associated to the following matrix:

$$\Delta_{\text{LSCM}} = \begin{pmatrix} 1+\delta & 0 & 0 & -\delta \\ 0 & 1+\delta & \delta & 0 \\ 0 & \delta & 1+\delta & 0 \\ -\delta & 0 & 0 & 1+\delta \end{pmatrix}$$

The parameter δ can be set to any positive value. We choose $\delta_{\text{LSCM}} = 10$ in our implementation.

Area distortion. For the matrix J to not deform areas, its determinant should be equal to 1. To take this into account, we add the term $2\delta(ad - bc)$ in the quadratic form by using the matrix:

$$\Delta_{\text{AREA}} = \begin{pmatrix} 1 & 0 & 0 & \delta \\ 0 & 1 & -\delta & 0 \\ 0 & -\delta & 1 & 0 \\ \delta & 0 & 0 & 1 \end{pmatrix}$$

where the parameter δ can also be set to any positive value. In practice, we choose $\delta_{\text{AREA}} = 1$.

Identity Distortion The identity distortion consists of being as close to $a = d = 1$ and $b = c = 0$ as possible. While the former cannot be translated in a quadratic form, the latter can be favored by augmenting the weight associated to b^2 and c^2 in the norm. The corresponding matrix is therefore the following diagonal matrix:

$$M_{\text{Id}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \delta & 0 & 0 \\ 0 & 0 & \delta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

with $\delta_{\text{Id}} > 1$. In practice, we take $\delta_{\text{Id}} = 5$.

Isometric Distortion Finally, the ARAP distortion requires that the matrix J is as close as possible to an orthogonal matrix. While the norm of the columns cannot be set to 1 using a quadratic form, we can nonetheless force the dot product $ac + bd$ of the two columns to be as small as possible. This metric alone provides locally distorted results since a rotation matrix of any angle is still a good candidate. To mitigate this effect, we combine the penalty $\delta(ac + bd)$ with the identity distortion above. This leads to the following symmetric matrix:

$$\Delta_{\text{ARAP}} = \begin{pmatrix} 1 & 0 & \delta & 0 \\ 0 & 1+\delta & 0 & \delta \\ \delta & 0 & 1+\delta & 0 \\ 0 & \delta & 0 & 1 \end{pmatrix}$$

which is positive definite for $\delta_{\text{ARAP}} < \frac{2}{\sqrt{5}-1} \approx 1.618$. We set $\delta_{\text{ARAP}} = 1.5$ in our implementation.

Linear Constraints

A third approach to distortion minimization in to not minimize an energy function nor penalizing certain directions of descent in the optimizer but to instead restrict the feasible domain to distortions $\mathcal{D}(J) < d$. While this is impractical because of non-linearities, this idea can be efficiently applied to the identity distortion described above (Equation (5.5)). Indeed, a feasible region for Jacobian matrices using this distortion can be expressed component-wise as the following linear inequalities over the coefficients:

$$\begin{pmatrix} 1-\varepsilon & -\varepsilon \\ -\varepsilon & 1-\varepsilon \end{pmatrix} < \begin{pmatrix} a & b \\ c & d \end{pmatrix} < \begin{pmatrix} 1+\varepsilon & \varepsilon \\ \varepsilon & 1+\varepsilon \end{pmatrix}. \qquad (\mathcal{D}_\varepsilon)$$

Therefore, instead of trying to solve Problem 5.4, we rewrite the distortion as an inequality constraint and solve:

$$\min_{\omega,v,J} \quad R(\omega, v, J) \qquad \text{s.t. } (\mathcal{B}), (\mathcal{D}_\varepsilon). \qquad (5.7)$$

This new formulation can still be solved using our Levenberg-Marquardt implementation, but since we do not know in advance the minimum value of ε for which a feasible solution exists, we start with a very small value and increase it when the optimization plateaus. In practice, we start with $\varepsilon = 10^{-3}$ and perform 30 increments of the same value so that the final ε is equal to 10. The problem is then optimized a final time without the constraint (i.e., $\varepsilon = +\infty$). It is worth noting that for $\varepsilon < 1/2$, Equation $(\mathcal{D}_\varepsilon)$ prevents determinants of Jacobian matrices to ever go negative, making the constraint of Equation $(\mathcal{I})$ redundant.

5.3 Results and Comparison

We now describe the various experiments performed with our face-based version of the algorithm. We report convergence results on the same database of models used in the previous Chapter, quantitative comparison with the vertex-based version as well as an evaluation of the different distortion minimization strategies exposed in the previous Section.

5.3.1 Database and Implementation

We use the same database of 274 triangular meshes, 143 of which have feature edges.

The implementation of the face-based algorithm mirrors as much as possible its vertex counterpart, and all experiments have been run using the same machine.

We report 16 failure cases for models with features and 4 for models without features, which is respectively 6 more and 4 less than the vertex-based results. This discrepancy can be explained by two factors. Firstly, the rotation's correction around sharp corners seems less effective in a face-based context, leading to failures on such models where pointy areas are degenerated which is in conflict with the injectivity constraint ($\mathcal{I}$). Secondly, a face-based approach using Jacobian matrices is less sensitive to triangles with extreme aspect ratios, thus allowing the algorithm to converge on models with bad triangles without the need for remeshing.

5.3.2 Comparison with a Vertex-based Implementation

We then provide a more in-depth comparison between the two versions of our algorithm. In Figure 5.4, we provide visual comparison between the results obtained on three models. We observe that without optimizing for a given distortion, the face-based algorithm tends to produce more singularities and more stretch distortion.

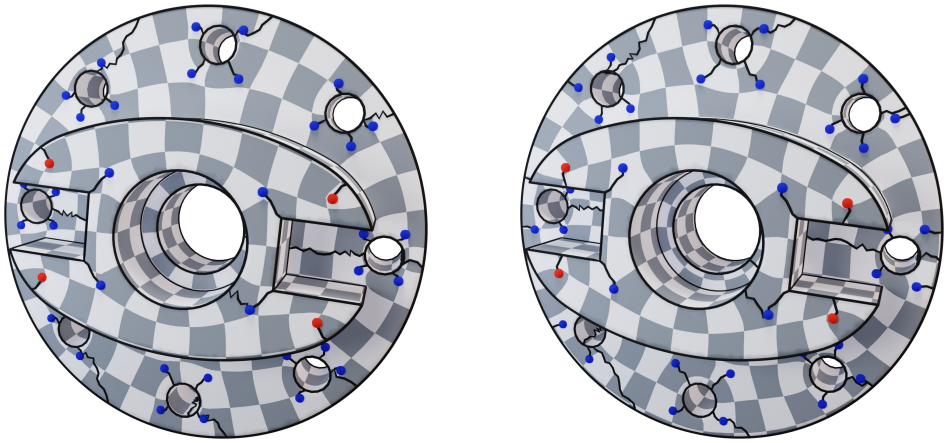
Computation time . As a triangle is described by a single chart matrix and not a collection of three vertex-based quad charts, the face-based discretization requires significantly fewer variables for the same input mesh. Indeed, the total number of floating point variables e, s, v, ω for the vertex-based discretization was $24|T| + 2|V| + |E|$, respectively the chart, frame field and rotation variables. In the face-based version, only $4|T| + 2|T| + |E|$ variables are required, for the triangle's Jacobian matrices, frames and rotation angles respectively. This has a direct impact on computation times : in Figure 5.5, we observe a speedup of around 30% for a majority of models, although a small amount performs actually worse for the face-based context. As for a complexity trend, the regression slope for the face-based version is smaller than its vertex counterpart, with $k_{\text{face}} = 1.2$ against $k_{\text{vertex}} = 1.3$, indicating a better performance on average.

Quality and convergence. We also run the database while minimizing each of the three distortion energies in common, namely $\mathcal{D}_{\text{LSCM}}$, $\mathcal{D}_{\text{ARAP}}$ and $\mathcal{D}_{\text{AREA}}$ along with no distortion at all. In Figure 5.6, we plot the *stretch* and *scale* distortion for each converged model in the dataset. We observe that for no distortion and for $\mathcal{D}_{\text{ARAP}}$, the two distributions are nearly indistinguishable, indicating very similar results. For $\mathcal{D}_{\text{LSCM}}$ and $\mathcal{D}_{\text{AREA}}$ on the other hand, the face-based version is able to better minimize stretch and scale respectively, hinting again at the fact that the face-based discretization better controls the distortion due to a lack of intrinsic flattening in initialization.

5.3.3 Distortion Strategies

Finally, we investigate the effectiveness of the different distortion minimization strategies exposed in the previous Section.

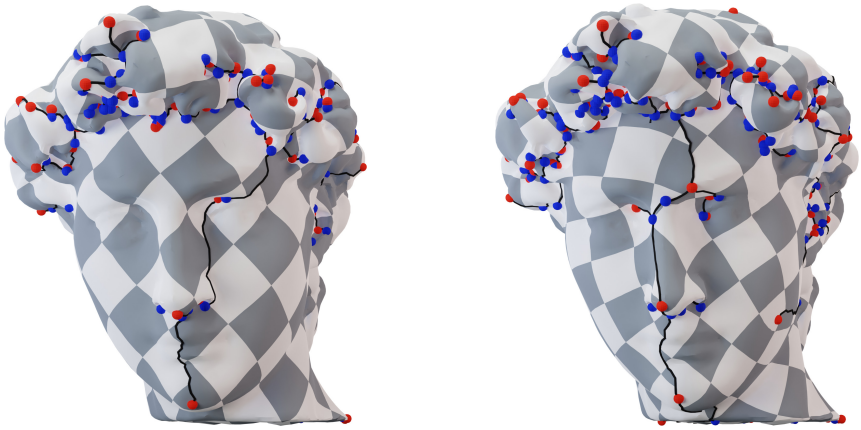
Least-squares vs metric change In Figure 5.8, we compare this distortion penalization to the energy minimization strategy in terms of *scale* and *stretch* metrics. Very similar results can be observed for the LSCM distortion in terms of stretch (y-axis) while the ARAP and



72 cones | Time = 453s
Scale = 1.032 | Stretch = 1.314

72 cones | Time = 316s
Scale = 1.031 | Stretch = 1.351

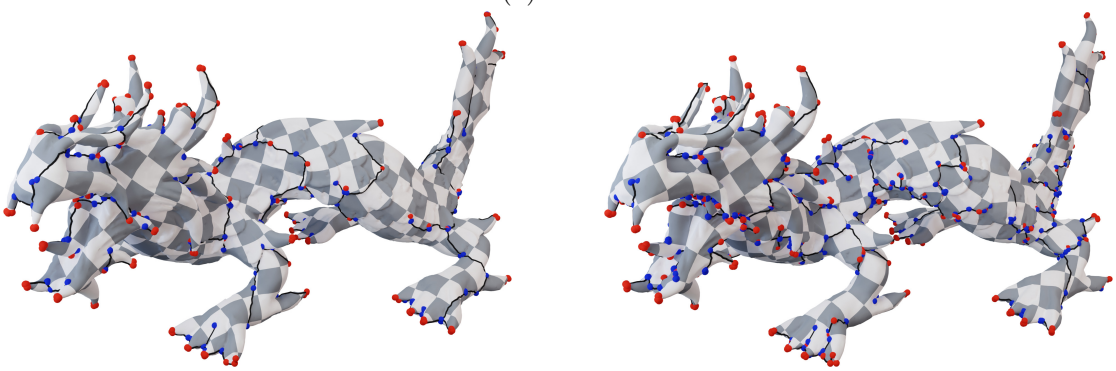
(a) M1



234 cones | Time = 833s
Scale = 1.011 | Stretch = 1.163

350 cones | Time = 449s
Scale = 1.018 | Stretch = 1.251

(b) David



582 cones | Time = 5073s
Scale = 1.013 | Stretch = 1.187

1159 cones | Time = 4989s
Scale = 1.015 | Stretch = 1.315

(c) Dragon

Figure 5.4: Comparison of the vertex-based version of the algorithm (left column) with the face-based version (right) on three models from the database. For each model, number of cones, distortion metrics and computation times are reported. The face-based version of the algorithm achieves similar performance with less computation time.

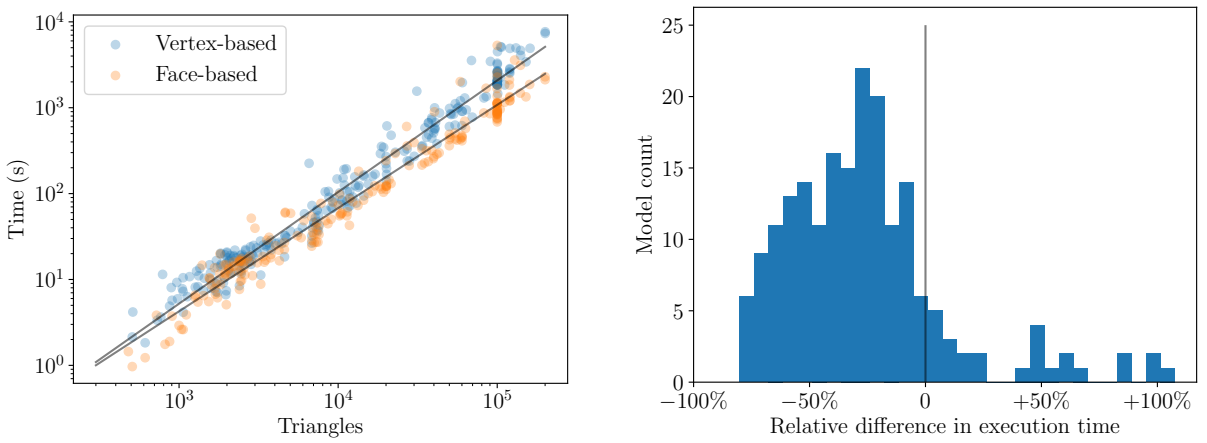


Figure 5.5: Comparison of running time between the face-based and the vertex-based implementations. Left: scatter plot of the running time with respect to the triangle count. Right: histogram of relative improvement of the face-based version. Linear regressions for both regressions show a very similar trend ($k = 1.2$ vs $k = 1.3$). For the majority of models, we observe that the face-based implementation is faster by 10 to 50%. For a small proportion of models, however, the optimization has to run for longer.

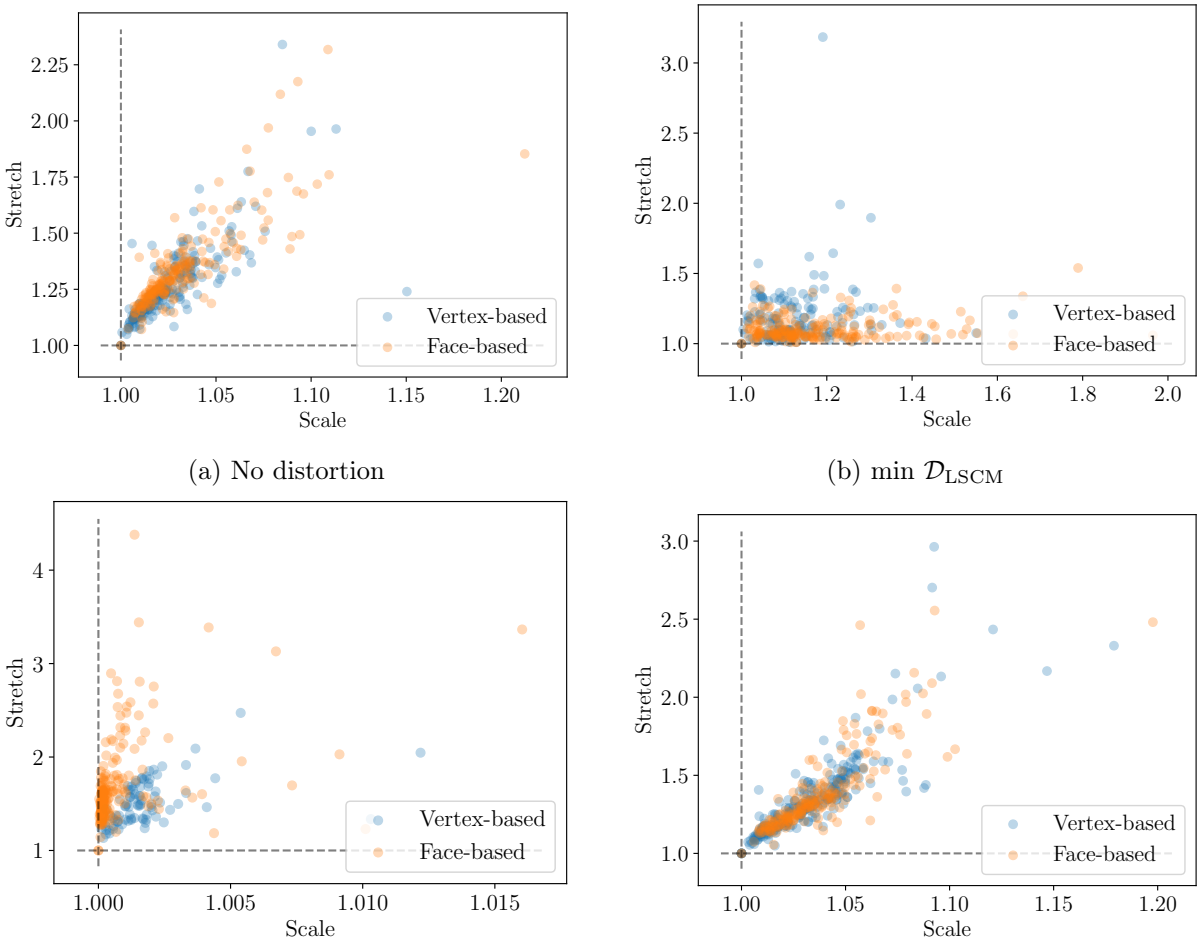
Id distortions converge to another slightly worse tradeoff. As far as the AREA distortion is concerned, results for the energy minimization strategy are significantly better in terms of scale (x-axis). We conclude that the metric change is especially effective to obtain more conformal parametrizations, but fails when it comes to authalic parametrizations, as preserving areas cannot be done linearly in practice.

Identity distortion Finally, in Figure 5.8d, we also plot the stretch and scale distortion for our third strategy, namely restraining the solution space using linear constraints. We observe that in this case, the resulting distortions are very similar to the ones reported for the energy minimization strategy, i.e., an expected tradeoff between scale and stretch. However, we report for this strategy a speedup of x2 to x5 of some models, due to the optimizer converging faster for each value of the hyperparameter ε.

5.4 Conclusion and Perspectives

Building on the same system of continuous equations stemming from Cartan's method of moving frames, we presented in this chapter a different discretization scheme based directly on the faces of a triangular mesh instead of vertex charts. The resulting optimization algorithm proves to be simpler to implement, involves less variables and runs significantly faster than its vertex-based counterpart in the majority of test cases. This would make a face-based discretization a go-to solution from a practical application of the method.

Thanks to the set of variables being smaller and representing more directly the parametrization's geometry, we overall gained much more freedom over the parametrization's geometry. We are able to explore alternative ways of minimizing a distortion metric, namely by tricking the optimizer's direction of descent towards less distorted elements, or using the simpler expression of distortion to enforce them as linear constraints. Yet, one could argue that this freedom gained



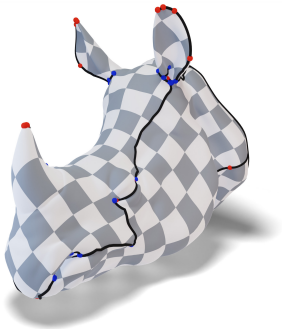
(a) No distortion

(b) min $\mathcal{D}_{\text{LSCM}}$

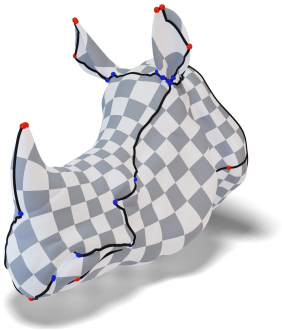
(c) min $\mathcal{D}_{\text{AREA}}$

(d) min $\mathcal{D}_{\text{ARAP}}$

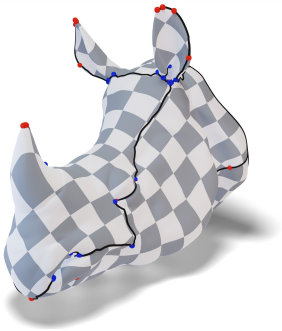
Figure 5.6: Scatter plot of the distortion obtained by the face-based and vertex-based versions of our algorithm, without minimizing distortion and for the three distortion energies in common : $\mathcal{D}_{\text{LSCM}}$, $\mathcal{D}_{\text{AREA}}$ and $\mathcal{D}_{\text{ARAP}}$. We observe very similar results for both versions without distortion and while minimizing $\mathcal{D}_{\text{ARAP}}$. For $\mathcal{D}_{\text{LSCM}}$ and $\mathcal{D}_{\text{AREA}}$, the face-based version is able to better minimize stretch (resp. scale) at the expense of more scale (resp. stretch).



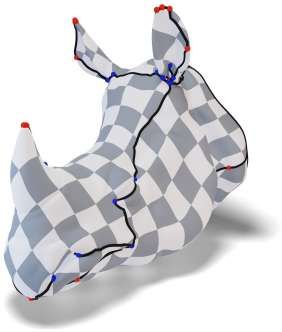
78 cones
Scale = 1.033
Stretch = 1.251
(a) No distortion



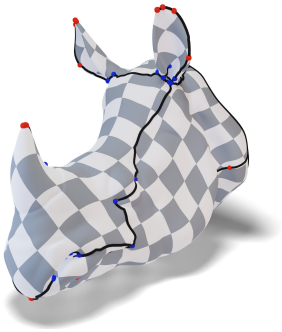
60 cones
Scale = 1.259
Stretch = **1.048**
(b) min $\mathcal{D}_{\text{LSCM}}$



74 cones
Scale = 1.037
Stretch = 1.284
(c) min $\mathcal{D}_{\text{ARAP}}$

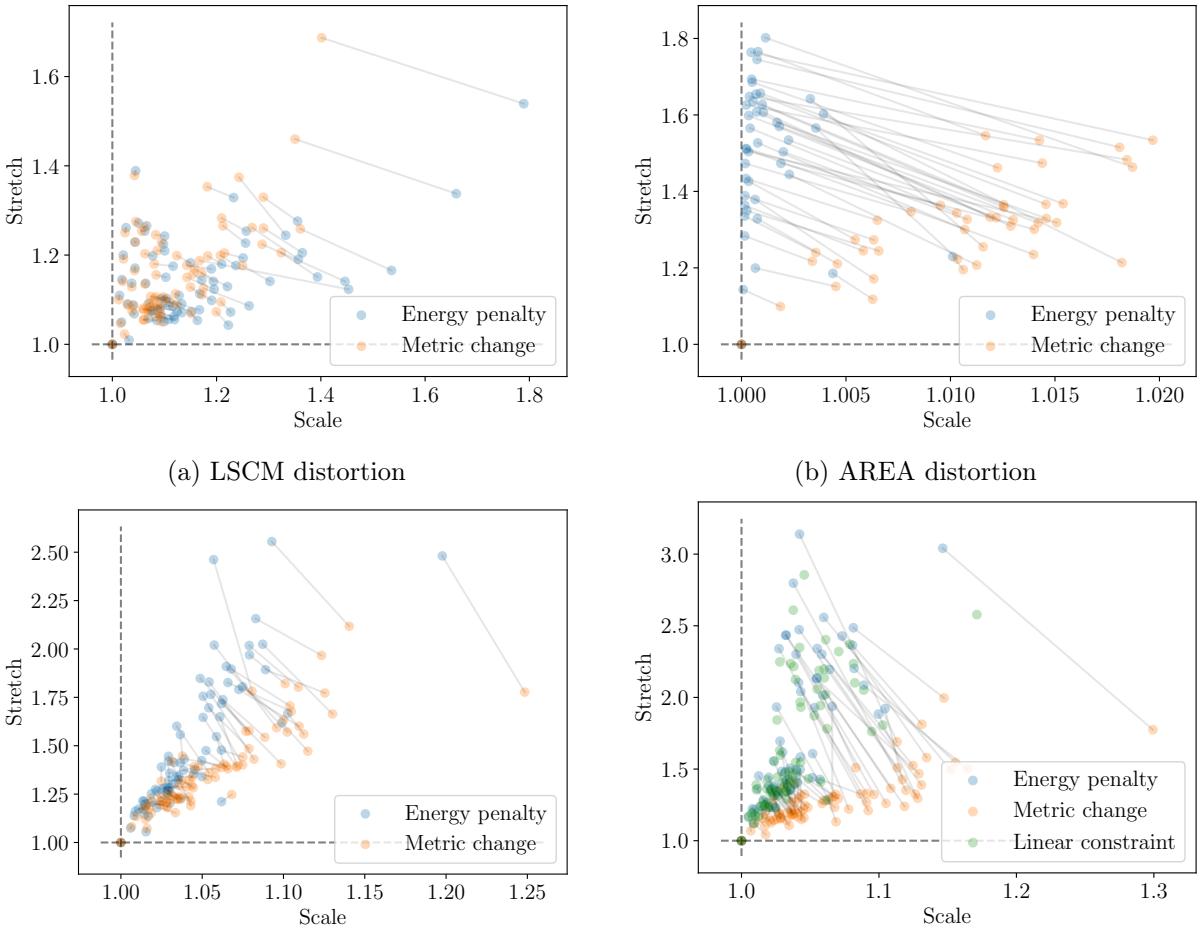


66 cones
Scale = 1.024
Stretch = 1.370
(d) min $\mathcal{D}_{\text{Id}}$



72 cones
Scale = **1.000**
Stretch = 1.510
(e) min $\mathcal{D}_{\text{AREA}}$

Figure 5.7: Parametrization of the *Rhino* head model minimizing the different distortion of the face-based algorithm. While $\mathcal{D}_{\text{LSCM}}$ and $\mathcal{D}_{\text{AREA}}$ minimize stretch and scale respectively, the two other distortions lead to a solution close to the one found without distortion. Results to be compared to Figure 4.1 for the vertex-based equivalent.



(a) LSCM distortion

(b) AREA distortion

(c) ARAP distortion

(d) Identity distortion

Figure 5.8: Scatter plot of the distortion obtained by the algorithm for the different strategies of distortion minimization. Points corresponding to the same model are linked by a gray line. For the first three distortions (LSCM, AREA and ARAP), we compare the energy minimization with the change of metric strategy. For the $\mathcal{D}_{\mathrm{Id}}$ energy, we also compare the linear constraint approach.

on the geometry is counterbalanced by the topology being more rigid, since singularity cones are bound to appear at vertices and not anywhere inside faces. In our experiments, this has a significant effect only on very coarse meshes, where the loss of degrees of freedom prevents a face-based version to converge. But where the number of triangles is sufficient, we observed that both discretizations exhibit the same behavior, meaning that forcing cones to appear at vertices do not prevent the algorithm from simultaneously finding a parametrization's geometry and topology.

More importantly, the practical benefits of a face-based approach come at the expense of theoretical guarantees on the correctness of the result. Indeed, double-coverings of vertex rings are no longer prevented and can happen in practice. This has been demonstrated on specifically crafted input meshes. However, with the exception of those specifically crafted examples, double-coverings do not appear in our experiments, so they can be considered unlikely.

Finally, whether it is face-based or vertex-based, the freedom that Cartan's method provides unfortunately requires a non-linear optimization, which is still orders of magnitude slower than any two-steps methods first computing a smooth frame field and then a parametrization via its integration. This is an expected price to pay for having conjoint control on the topology and the geometry. Yet, due to the form of the energy functions considered, only comparing adjacent charts in the mesh in a "Laplacian" style, we believe that improvement on performance is still possible both in implementation details and thanks to a better conditioning of the optimizer.

6

Grid-preserving Parametrization in a Single Optimization

Contents

6.1 Periodic Global Parametrization 138
6.1.1 Principle 138
6.1.2 Algorithm 140
6.1.3 Behavior and Limitations 142
6.2 Motivations 143
6.3 Seam-agnostic Oscillator Functions 144
6.3.1 Symmetry Sets and Functional Representation 144
6.3.2 Decomposition in Fourier Basis 144
6.3.3 Oscillator Design 149
6.4 Parametrization Setup 150
6.4.1 Setup per Triangle 150
6.4.2 Feature Constraints 150
6.4.3 Adjacency Constraint 151
6.4.4 Injectivity and Distortion 152
6.4.5 Oscillators Validity 152
6.4.6 Final optimization problem 153
6.5 Conclusion 153

Computing a quadmesh of a given shape using a parametrization-based approach consists in solving all the steps of the pipeline we exposed in Section 2.4.2 of Chapter 2. In short, since directly computing a constrained parametrization suited for quad extraction is difficult, a common approach is to compute the different sets of degrees of freedom one after the other using more tractable optimization problems. The first step is to determine position of singularity cones. Then, the rotational part of the transition functions are computed at seams to obtain a seamless parametrization. Finally, a quantization step aligns every feature point to the integer grid and thus determines integer translations across seams.

In the two previous chapters, we have described a method to compute a seamless parametrization of an arbitrary surface that solves simultaneously for both a singularity distribution and a low distortion parametrization. In a sense, this method of moving frames solves the first two

steps of the pipeline in a single optimization. Although this optimization problem is non-convex, we demonstrated the versatility of our approach in practice as well as the control it provides over the resulting parametrization.

However, using our algorithm to compute a quadmesh still requires an additional quantization procedure to determine the integer translations across seams. This second step is susceptible to introduce additional distortion and singularities not accounted for in the method of moving frames, which in a way negates our efforts for computing the optimal distribution of singularities for a given distortion metric.

In this chapter, we explore the possibility of extending our previous work to take the quantization step into account. In other words, we attempt at designing an optimization algorithm that, given a triangle mesh of shape with prescribed feature edges, will compute an *integer* seamless parametrization, solving for all degrees of freedom simultaneously. As such an algorithm would not suffer from failures due to irreversible suboptimal choices made in a pipeline of several steps, our ultimate hope is to unlock full control over the resulting quadmesh.

This approach is challenging because of the different nature of the degrees of freedom needed to be solved simultaneously: singularity positions and rotationally seamless maps can be computed using continuous optimization, sometimes as simple as a linear least-square, whereas conditioning a parametrization to be grid-preserving involves a set of integer variables, namely the translations' coordinates at seams. These mixed-integer problems are notoriously NP-hard in the general case, thus rendering this formulation prohibitive for an "all-in-one" optimization.

To tackle this issue, we draw our inspiration from a method that avoids the use of mixed-integer problems: the *Periodic Global Parametrization* (PGP) algorithm. Introduced by Ray et al. [2006] and already discussed briefly in Section 2.4.5, PGP utilizes periodic functions of the uv-coordinates, called oscillator functions, to optimize a parametrization up to any integer translation. The resulting equations are expressed as a linear least-square problem and can therefore be solved efficiently. The main difficulty in our approach is that PGP is initialized using an approximate parametrization where singularity positions and seam constraints are already known. In our case, we aim at designing a variant of PGP that is agnostic of these information in order to freely add or remove singularities.

This chapter is organized as follows. We start by describing the original PGP algorithm and its behavior in Section 6.1. In Section 6.2, we motivate our approach for an improved PGP algorithm. In Section 6.3, we design a different oscillator function that allows a PGP-like system to be agnostic of seams and singularities positions. Finally, in Section 6.4, we expose an energy function depending on local deformations and those oscillator functions whose optimization implies the computation of a grid-preserving parametrization. As the method discussed in this chapter is an ongoing work, it has yet to be validated by practical results.

6.1 Periodic Global Parametrization

Let us begin by describing the original *Periodic Global Parametrization* algorithm introduced by Ray et al. [2006].

6.1.1 Principle

Given a triangle mesh $M = (V, E, T)$ and a parametrization ϕ that acts as a good initial guess, PGP computes a grid-preserving parametrization from which quads can be directly extracted. Recall (from Definition 2.4 in Section 2.4.2) that a parametrization is grid-preserving if uv-coordinates across a seam relate to one another by a square-preserving rotation and an integer

translation. In other words, for any two points (u, v) and (u', v') in parameter space corresponding to the same vertex, there exists integers k, n_u, n_v such that:

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = R \left(\frac{\pi}{2}\right)^k \begin{pmatrix} u + n_u \\ v + n_v \end{pmatrix}. \tag{6.1}$$

The difficulty of finding uv-coordinates satisfying this equation lies in the integer nature of the translation, which naturally calls for hard optimization problems over integer variables. The idea behind the PGP algorithm is to not work directly with uv-coordinates but to consider periodic functions over the domain, which we will be referring to as *oscillators*. For a point (u, v), we define its oscillator as:

$$\theta(u, v) := \begin{pmatrix} \cos(u) \\ \sin(u) \\ \cos(v) \\ \sin(v) \end{pmatrix}. \tag{6.2}$$

To account for the fact that the cos and sin functions are 2π-periodic and not 1-periodic, we will consider our uv-coordinates to be globally scaled by a factor 2π and work with values in $2\pi\mathbb{Z}$ instead of just $\mathbb{Z}$. All equations will be equivalent up to this global scale.

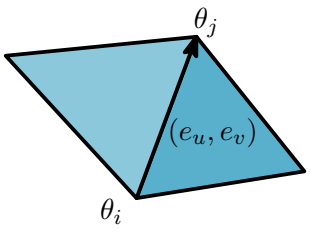
An important property of the oscillators is that $\theta(u + n_u, v + n_v) = \theta(u, v)$ for any number $n_u, n_v \in 2\pi\mathbb{Z}$. As a consequence, expressing Equation (6.1) in terms of oscillators make the integer translation disappear, leaving only a linear relationship over seams:

$$\theta(u', v') = Q^k \theta(u, v) \quad \text{where } Q = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}. \tag{6.3}$$

The goal of PGP is to therefore work with these indirect variables defined at each triangle corner of a parametrization in order to satisfy Equation (6.3) at each edge and recover a parametrization that is grid-preserving by construction. For this, however, it has to ensure that there exist underlying functions u and v such that any $\theta_c \in \mathbb{R}^4$ defined at a triangle corner c is indeed of the form $\theta(u_c, v_c)$. This condition can be achieved in two steps.

Firstly, a parametrization from which quads can be extracted is a cone parametrization whose cones all have integer coordinates and feature edges are all vertical and horizontal. In other words, for any corner c that is a cone of the parametrization, the value of θ_c is fixed: $\theta_c = (1\ 0\ 1\ 0)^T$. And for corners on a feature edge, we want either that $\theta_c = (1\ 0\ .\ .)^T$ or $(.\ .\ 1\ 0)^T$.

Secondly, this information can be propagated to other points by considering translations in parameter space. If two points (u_i, v_i) and (u_j, v_j) are linked in parameter space by an edge $e = (e_u, e_v)$ (inset Figure), then we can simply write:



$$\begin{pmatrix} \cos(u_j) \\ \sin(u_j) \\ \cos(v_j) \\ \sin(v_j) \end{pmatrix} = \begin{pmatrix} \cos(u_i + e_u) \\ \sin(u_i + e_u) \\ \cos(v_i + e_v) \\ \sin(v_i + e_v) \end{pmatrix}$$

which can be developed into a linear relationship between the corresponding oscillators θ_i and θ_j as:

$$\theta(u_j, v_j) = \hat{R}(e_u, e_v)\theta(u_i, v_i) \tag{6.4}$$

where:

$$\hat{R}(e_u, e_v) = \begin{pmatrix} R(e_u) & 0 \\ 0 & R(e_v) \end{pmatrix} \quad \text{and} \quad R(x) = \begin{pmatrix} \cos(x) & -\sin(x) \\ \sin(x) & \cos(x) \end{pmatrix}.$$

In other words, oscillators from two different points p_i and p_j in parameter space are related by a block diagonal rotation matrix whose two angles are exactly the coordinates of the translation that aligns p_i to p_j.

6.1.2 Algorithm

Thanks to Equations (6.3) and (6.4), we have all the relations we need to describe the PGP method. Overall, PGP defines oscillator functions θ as vectors of $\mathbb{R}^4$ at corners of a triangle mesh and solves Equation (6.4) for each half-edge of the mesh given their coordinates (e_u, e_v) in parameter space. This involves a linear-least square with linear constraints formulation where the sum is taken for every edge of every triangle in the mesh:

$$\begin{aligned} &\min_{\theta} \sum_{t \in T} \sum_{e^t=(a,b) \in t} ||\theta_b^t - \hat{R}(e_u^t, e_v^t)\theta_a^t||^2 \\ \text{Subject to: } &\theta_s = (1\ 0\ 1\ 0)^T \text{ for all singularities } s \\ &\theta_c = (1\ 0\ \textbf{.}\ \textbf{.})^T \text{ or } (\textbf{.}\ \textbf{.}\ 1\ 0)^T \text{ for corners } c \text{ on a feature} \\ &\theta_{c'} = Q^k \theta_c \text{ for } (c, c') \text{ corresponding across a seam with jump index } k. \end{aligned} \tag{6.5}$$

Having oscillators at corners in this formulation introduces many variable redundancies. In practice, the number of variables can be greatly reduced by observing that for edges with jump index $k_e = 0$, the seam constraint of Equation (6.3) calls for a simple equality between oscillators. By defining oscillator functions at the vertices of a transformed mesh where seam edges ($k_e \neq 0$) have been explicitly cut, one avoids many equations in Problem (6.5). By doing so, we are left with $n + 1$ oscillator functions for a vertex v whose neighborhood contains n cuts.

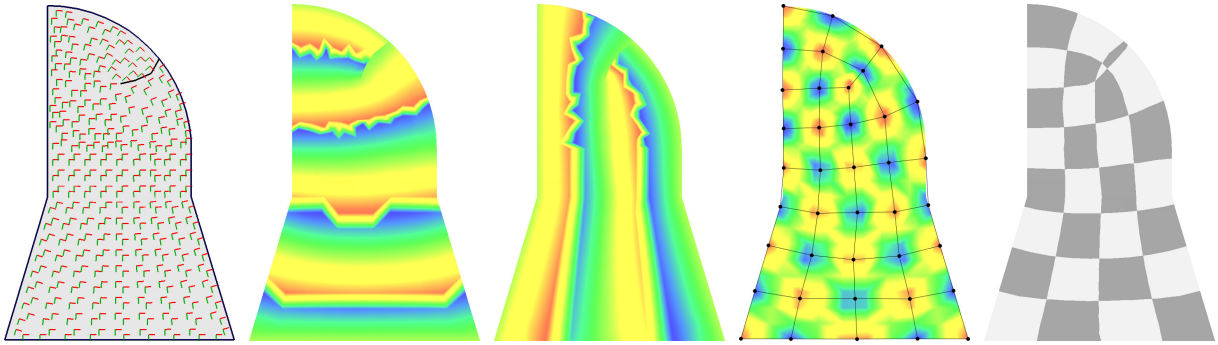
Since the solution of Problem (6.5) is only a least-square approximation of the constraint to be satisfied, there is no guarantee that underlying functions u and v such that oscillators θ have the form $\theta(u, v) = (\cos(u)\ \sin(u)\ \cos(v)\ \sin(v))^T$ exist. In practice, a projection to the nearest valid oscillator can be performed after solving Problem (6.5). The arguments $\hat{u}$ and $\hat{v}$ of these oscillators are shown in Figure 6.1b and 6.1c.

Several issues need to be addressed in order to solve Problem (6.5), both at the initialization step and at the quad extraction step.

Initialization In order to solve Problem (6.5), one first needs to define the rotation matrices $R(e_u, e_v)$ for each edge. This is done by providing a "good guess" of what the edge coordinates (e_u, e_v) are in parameter space. In practice, a smooth frame field is computed on the input triangle mesh, from which two orthogonal vector fields z_u and z_v are extracted (Figure 6.1a). These vector fields form discontinuities at seams and singularities that completely determine the jump indices k present in Equation (6.3), as well as alignment of feature edges with either u or v.

Then, candidate edge coordinates (e_u, e_v) are computed from the frame field by projecting the edge coordinates from xyz-space to the basis formed by z_u and z_v (Figure 6.1a):

$$\begin{pmatrix} e_u \\ e_v \end{pmatrix} = \begin{pmatrix} (e_x, e_y, e_z) \cdot z_u \\ (e_x, e_y, e_z) \cdot z_v \end{pmatrix}.$$



(a) (b) (c) (d) (e)

Figure 6.1: Illustration of the different steps of the PGP algorithm on a simple model. (a) Initialization: a smooth frame field is computed and two orthogonal vector fields (red and green) are extracted, as well as seams and singularities (no curl correction). (b) and (c) Arguments of the u and v parts of the oscillator θ after optimization. (d) Plot of the function $\theta^{(0)}\theta^{(2)} = \cos(u)\cos(v)$ and its quasi-dual Morse-Smale complex, forming a valid quadrangulation. (e) Extracted grid-preserving uv-coordinates.

In general, there is no guarantee that vector fields z_u and z_v are gradient fields. Since they are of constant unit norm, the resulting parametrization tends to be isometric with many irregularities (like T-junctions, as will be discussed in Section 6.1.3). In order to regularize the result, the authors of PGP describe a *curl correction* step where the initial vector fields are multiplied by a factor $\lambda > 0$ chosen so that their curl is minimized. This amounts at scaling the corresponding edge coordinates by the same factor λ.

Quad Extraction From the solution of Problem (6.5), the authors compute uv-coordinates from oscillators:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \arctan(\theta[1]/\theta[0]) \\ \arctan(\theta[3]/\theta[2]) \end{pmatrix}$$

where we denote by $\theta[i]$ the i-th coefficient of vector θ.

When using this formula, many issues remain to be fixed in order to get a valid quadmesh: global translations of 2π need to be addressed while inconsistencies in lengths (see Section 6.1.3) can appear and require additional refinement or remeshing.

Reconstruction via Morse-Smale complex Another approach to extract quads is described by Zhang et al. [2010]. The idea consists instead in computing a smooth oscillating function over the domain:

$$f(u, v) = \cos(u)\cos(v) = \theta[0]\theta[2]$$

and extract its *quasi-dual Morse-Smale complex*. In general, the Morse-Smale complex of a function is a graph that links minima and maxima to their adjacent saddle point Banchoff [1970]. This graph can be shown to always be a quadrangulation in the case of surfaces [Dong et al. 2006]. In the case of the PGP function, these quadrangles are unfortunately rotated by $\pi/4$, but the quasi-dual Morse-Smale complex, linking diagonally adjacent maxima and minima, provides a good quadrangulation. This is illustrated in Figure 6.1d.

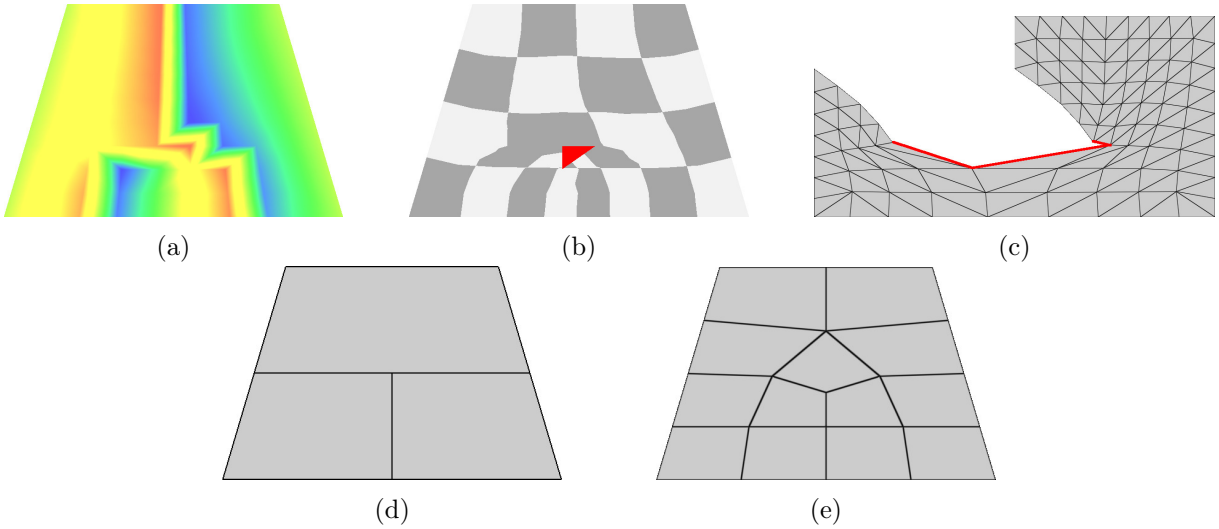


Figure 6.2: A T-junction appears in PGP when the number of oscillations is inconsistent between two feature lines. This is illustrated here on a trapezoid. (a) The u part of the oscillator makes one total oscillation on the top boundary and two on the bottom. (b) This results in a parametrization with twice as much length on the bottom as well as a broken triangle (red) that cannot be reconstructed in parameter space. (c) Visualization of the uv-coordinates with the edges of the broken triangle highlighted in red. (d) Locally extracting quads following Ray et al. [2006] allows the creation of a T-mesh. (e) A valid quad mesh can be computed using a subdivision scheme like the Catmull-Clark subdivision, leading to a mesh with a dipole of singularities.

6.1.3 Behavior and Limitations

The PGP algorithm is a radically different approach to quadmeshing than the classical parametrization-based pipeline, which can be both a strength and a drawback. One of its key behaviors is indeed its ability to add singularities to account for incompatible lengths in the resulting parametrization. This property is illustrated in Figure 6.2 on the simplest possible example: the trapezoid. Since the oscillator functions optimized in PGP are periodic, finding a solution of Problem (6.5) implies that the length in parameter space allows the oscillators to perform an integer number of periods between vertices of interests (in the case of this trapezoid: the four corners), but this comes at the cost of not knowing exactly the value of this integer, which usually ends up being a simple rounding of the considered length. For our trapezoid, this rounded length turns out to be 1 on the top edge and 2 on the bottom edge, as shown in Figure 6.2b. This leads to incoherence in the parametrization in the form of a broken triangle (red in Figure 6.2b) whose three edges do not form a valid triangle in parameter space (red highlights in Figure 6.2c). This means that no valid grid-preserving parametrization can be extracted, even in this simple case. However, using the quad extraction procedure of Ray et al. [2006], it is still possible to extract quads locally and form a T-mesh like in Figure 6.2d. Fortunately, a valid quad mesh can always be extracted from this kind of T-mesh using a subdivision algorithm. For instance, in Figure 6.2e, we apply the Catmull-Clark subdivision [Catmull and Clark 1998] on the T-mesh to retrieve the final result, but this introduces two new singularities that should have appeared inside the broken triangle.

In summary, PGP is able to produce a valid quadmesh by adding pairs of singularities

(dipoles) every time the oscillators converge to inconsistent lengths. This is a key behavior of the method but comes at the cost of an invalid underlying parametrization and requires an indirect extraction of the quads. Moreover, getting a valid quadmesh from the resulting T-mesh may require many subdivision which increases the number of elements of the mesh, preventing the computation of a coarse block decomposition [Bommes et al. 2013a].

In addition, only little control is given by the algorithm over the number and the position of the additional dipoles. Only the curl correction preprocessing step allows for some lengths to be scaled in the initial parametrization, fixing some of the encountered inconsistencies, but it does not completely prevent additional dipoles from appearing.

Finally, the relevance of having an algorithm that is able to add dipoles of singularities at this point of the pipeline can be questioned. Since frame field methods tend to underestimate the number of singularities necessary for a quad mesh to be of good quality (they focus on producing the smoothest possible field), it can be desirable that the following step is able to compensate for this and introduce the remaining irregular points of the final quadmesh. But in a context where the previous step aimed at computing the best possible singularity distribution, as we attempted in the previous chapters, it would be preferable to either compute a quantization with fixed topology or having full control over the singularity distribution, that is being able to add but also *remove* singularities.

6.2 Motivations

Now that we have discussed the original PGP algorithm, let us motivate our need for a new algorithm directly computing grid-preserving parametrizations.

As we saw in the previous section, PGP does not always produce a valid parametrization but instead creates T-junctions and broken triangles. We would instead like to obtain an algorithm that maps each triangle to a triangle in parameter space. However, the ability to break triangles and thus place additional singularities is what allows PGP to converge and find a solution. Preventing triangles from breaking therefore needs to be compensated by the ability to freely place singularities at vertices. This requirement has two consequences. Firstly, we need an algorithm that does not rely on the knowledge of initial singularities or seam, but discovers a singularity distribution during optimization. This calls for new oscillator functions that take eventual square-preserving rotations as well as integer translations into account. Secondly, as we would like the singularities to change depending on distortion criteria, we need to also optimize for the parametrization's edge coordinates along the oscillators, leading to a non-linear optimization approach.

By considering both the parametrization's edge coordinates and a set of seam-agnostic oscillator functions, we will be able to exploit the known tradeoff between the number of singularities and the distortion of the parametrization. There are indeed two types of solutions to be expected from any algorithm computing a grid-preserving parametrization from a seamless one. The first one is to keep the exact same set of singularities and modify lengths. This is the solution usually produced by mixed-integer approaches like the ones from Bommes et al. [2009] or Campen et al. [2015]. The second one is to conserve lengths and instead rely on additional singularities to achieve coherence. This is what the PGP algorithm does. To the best of our knowledge, no method has been able to exploit the tradeoff between these two outputs correctly. As we did with the method of moving frames in Chapters 4 and 5, our goal is to control local distortion as a mean of adding compensating singularities.

6.3 Seam-agnostic Oscillator Functions

Now that our motivations have been stated, let us focus on the description of an algorithm that satisfies them all. The first step is to design novel oscillator functions ψ that still enforce the grid-preservation property for all edges even when seams of the parametrization are not known in advance. To do so, we describe a general method to design representation of objects for them to be smoothly optimized up to a desired symmetry. This approach is inspired by Desobry et al. [2021] where it was applied to the problem of non-orthogonal frame fields.

6.3.1 Symmetry Sets and Functional Representation

Let us recall the desirable properties we need over the uv-coordinates. Given two corners (u, v) and (u', v') corresponding to each other across an edge, we want that their uv-coordinates satisfy Equation (6.1). Put differently, we can define the equivalence class of a point (u, v) as all the points that are reachable by application of the considered symmetries, namely:

$$\mathcal{C}_{u,v} := \left\{ R\left(\frac{\pi}{2}\right)^k \begin{pmatrix} u + n_u \\ v + n_v \end{pmatrix} \mid k, n_u, n_v \in \mathbb{Z} \right\}$$

Using this definition, we simply require that $\mathcal{C}_{u,v} = \mathcal{C}_{u',v'}$ for each corresponding pairs of points across an edge. Matching those infinite sets is, of course, intractable in practice. The trick is to instead work in a vector space of functions $\mathcal{F}$ such that each set $\mathcal{C}_{u,v}$ corresponds to a unique function $f_{u,v} \in \mathcal{F}$. Given a chosen basis of $\mathcal{F}$, one can then represent $f_{u,v}$ as its coordinate vector in the basis and optimize over this set of variables.

To design an approximate functional space $\mathcal{F}$, one has to consider functions with the same symmetries as the problem. As the latter depends on two parameters u and v, the functions should be of two variables x and y, written in the form $f(u, v; x, y)$. Imposing the grid-preserving symmetries onto those objects means that f should be periodic and invariant by $\pi/2$ rotation of its (u, v) parameters:

$$\left| \begin{array}{rcl} f(u, v; x, y) & = & f(u + n_u, v + n_v; x, y) \text{ for all } n_u, n_v \in 2\pi\mathbb{Z} \\ f(u, v; x, y) & = & f(v, -u; x, y) = f(-u, -v; x, y) = f(-v, u; x, y) \end{array} \right.$$

Many different functions satisfy those two conditions. One simple and straightforward choice is to consider the function:

$$g(u, v, x, y) = \cos(u - x)^{2d} \cos(v - y)^{2d}$$

for a sufficiently large $d \in \mathbb{N}$ and build $f(u, v; x, y)$ as:

$$f(u, v; x, y) = g(u, v, x, y) + g(-v, u, x, y) + g(-u, -v, x, y) + g(-v, u, x, y). \tag{6.6}$$

By construction, functions $f(u, v; x, y)$ satisfies the two symmetries we are looking for. Figure 6.3 plots the heightmap of f for a full x, y period of 2π and for different values of the parameters u and v. One can verify empirically that two points (t1) and (t4) related by a rotation of $\pi/2$ are associated with the same exact function.

6.3.2 Decomposition in Fourier Basis

We have now build a family of functions on which equality captures the equality up to the symmetries we want on the uv-coordinates. In order to work with these functions in practice, we

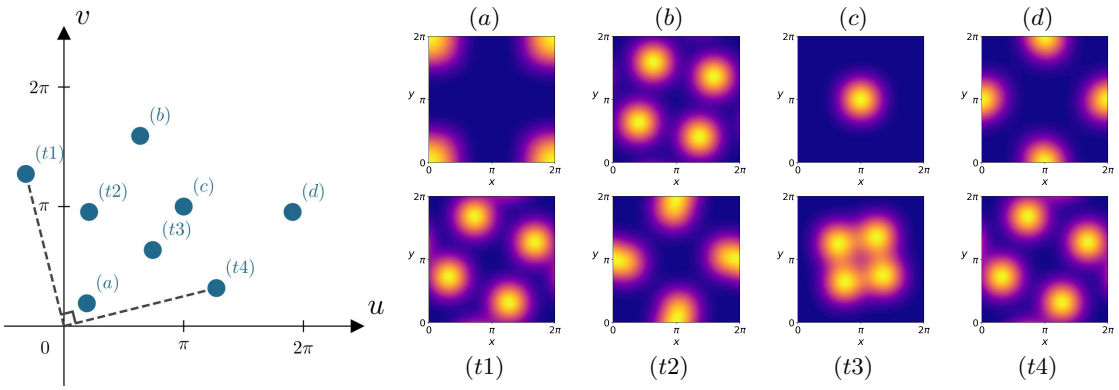


Figure 6.3: Visualization of the function $f(u, v; x, y)$ over the domain $[0; 2\pi]^2$ for different values of u and v. When (u, v) are in no particularly symmetrical position, the function exhibits four local maxima corresponding to the four summed versions of g. When symmetries occur, those maxima merge, like at the point $(c) = (\pi, \pi)$. On the bottom row, the four values are taken as linear interpolation between two points $(t1)$ and $(t4)$ that relate to each other by a rotation of $\pi/2$. We observe as expected that the functions are equal at $(t1)$ and $(t4)$.

represent them as their vector of coefficients in a given functional basis. Since we are working with periodic functions, a natural choice is to consider the two-dimensional Fourier basis. Indeed, for any values of parameters (u, v), there exists a set of coefficients $\{a_n, b_n, c_n, d_n | n \in \mathbb{N}\} \subset \mathbb{R}$ such that:

$$\begin{aligned} f(u, v; x, y) = &\sum_n a_n(u, v) \cos(nx) \cos(ny) + \sum_n b_n(u, v) \cos(nx) \sin(ny) \\ &+ \sum_n c_n(u, v) \sin(nx) \cos(ny) + \sum_n d_n(u, v) \sin(nx) \sin(ny) \end{aligned}$$

With the coefficients a_n, b_n, c_n, d_n being functions of u and v. Since f is a polynomial in cosine, its set of non-zero Fourier coefficients is finite and can be easily solved by computing the integral through which they are defined:

$$a_n(u, v) := \iint_0^{2\pi} f(u, v; x, y) \cos(nx) \cos(ny) dx dy.$$

It turns out that only a few coefficients are non-zero:

$$a_n, b_n, c_n, d_n \propto \begin{cases} \cos(ku) + \cos(lv) \\ \cos(ku) \cos(lv) \\ \cos(ku) \cos(lv) + \cos(lu) \cos(kv) \\ \sin(ku) \sin(lv) - \sin(lu) \sin(kv) \end{cases} \quad \text{for } 1 \leqslant k, l \leqslant 2d. \tag{6.7}$$

Since the Fourier basis is an orthogonal basis of $\mathcal{F}$, we know that equality between two functions is equivalent to the equality of all their coefficients in the basis. In other words, it is sufficient to consider the vector $(a_n, b_n, c_n, d_n)_{1 \leqslant n \leqslant 2d}$ of coefficients to represent a function and optimize for the L^2 norm between vectors in order to optimize for a distance between functions.

However, the number of distinct coefficient increases with the degree d and it can quickly become impractical to work with this many variables. In our case, we only need to find the minimal

d such that set equality between $\mathcal{C}_{u,v}$ and functional equality between $f(u,v)$ are equivalent. Fortunately, one can notice that equality between *all* coefficients is sufficient for functional equality, but not at all necessary. For instance, the product $\cos(ku)\cos(lv)$ can be expressed solely using the two sums $\cos(ku)+\cos(lv)$ and $\cos(2ku)+\cos(2kv)$, so that equality for this coefficient does not provide any new information.

Based on more trigonometric identities and instantiation for different values of k, l, we extract a sufficient set of coefficients whose matching between two points (u,v) and (u',v') still leads to the desired symmetries:

$$\begin{cases} \cos(u)+\cos(v) \\ \cos(2u)+\cos(2v) \\ \cos(\ u+2v)+\cos(2u-\ v) \\ \cos(2u+4v)+\cos(4u-2v) \end{cases}. \tag{6.8}$$

This fixes the minimal degree d to be 4.

We are now going to prove that matching those four coefficients between corresponding corners across an edge is indeed sufficient. By construction, we already know that if two points (u,v) and (u',v') relate to each other by a grid-preserving symmetry (i.e., $\mathcal{C}_{u,v} = \mathcal{C}_{u',v'}$), then equality of the coefficients in Equation (6.8) is verified. Let us turn our attention to the converse theorem:

Theorem 6.1. *Suppose that coefficients of Equation* (6.8) *are equal for two points* (u,v) *and* (u',v'). *Then* $\mathcal{C}_{u,v} = \mathcal{C}_{u',v'}$

Proof. To prove this result, we start by proving a simple lemma:

Lemma 6.2. *Let* $a,b,c,d \in \mathbb{R}$.

$$\begin{cases} a+b=c+d \\ a^2+b^2=c^2+d^2 \end{cases} \Rightarrow \begin{cases} a=c \\ b=d \end{cases} \text{ or } \begin{cases} a=d \\ b=c \end{cases}$$

The intuition behind this lemma is to look at a geometric interpretation of the two equations. The first one means that parameters a,b and c,d describe the same line in the plane. The second mean that they describe the same circle. Solutions of this problem are therefore intersection points between the line and the circle, of which there are at most two, so the two immediate solutions $(a,b)=(c,d)$ or (d,c) are the only possible ones.

Back to the proof of our theorem, suppose that the four following equations hold for our two points (u,v) and (u',v'):

$$\begin{cases} \cos(u)+\cos(v) &=& \cos(u')+\cos(v') \\ \cos(2u)+\cos(2v) &=& \cos(2u')+\cos(2v') \\ \cos(u+2v)+\cos(2u-v) &=& \cos(u'+2v')+\cos(2u'-v') \\ \cos(2u+4v)+\cos(4u-2v) &=& \cos(2u'+4v')+\cos(4u'-2v') \end{cases} \tag{6.9}$$

Using the fact that $\cos(2x) = 2\cos(x)^2 - 1$, System 6.9 is equivalent to:

$$\begin{cases} \cos(u)+\cos(v) &=& \cos(u')+\cos(v') \\ \cos(u)^2+\cos(v)^2 &=& \cos(u')^2+\cos(v')^2 \\ \cos(u+2v)+\cos(2u-v) &=& \cos(u'+2v')+\cos(2u'-v') \\ \cos(u+2v)^2+\cos(2u-v)^2 &=& \cos(u'+2v')^2+\cos(2u'-v')^2 \end{cases} \tag{6.10}$$

This puts the four equations into two pairs on which we can apply Lemma 6.2. The proof can then be divided into four cases:

$$\text{Case 1:} \begin{cases} \cos(u) = \cos(u') \\ \cos(v) = \cos(v') \\ \cos(u+2v) = \cos(u'+2v') \\ \cos(2u-v) = \cos(2u'-v') \end{cases} \qquad \text{Case 2:} \begin{cases} \cos(u) = \cos(u') \\ \cos(v) = \cos(v') \\ \cos(u+2v) = \cos(2u'-v') \\ \cos(2u-v) = \cos(u'+2v') \end{cases}$$

$$\text{Case 3:} \begin{cases} \cos(u) = \cos(v') \\ \cos(v) = \cos(u') \\ \cos(u+2v) = \cos(u'+2v') \\ \cos(2u-v) = \cos(2u'-v') \end{cases} \qquad \text{Case 4:} \begin{cases} \cos(u) = \cos(v') \\ \cos(v) = \cos(u') \\ \cos(u+2v) = \cos(2u'-v') \\ \cos(2u-v) = \cos(u'+2v') \end{cases}$$

For each case, the proof relies on developing the last two equations using the following identities:

$$\left| \begin{array}{rcl} \cos(x+2y) & = & \cos(x)\cos(2y) - 2\cos(y)\sin(x)\sin(y) \\ \cos(2x-y) & = & \cos(2x)\cos(y) + 2\cos(x)\sin(x)\sin(y) \end{array} \right. \tag{6.11}$$

Case 1. Application of Equations 6.11 and substitution leads to:

$$\begin{cases} \cos(u) = \cos(u') \\ \cos(v) = \cos(v') \\ \cos(v)\sin(u)\sin(v) = \cos(v')\sin(u')\sin(v') \\ \cos(u)\sin(u)\sin(v) = \cos(u')\sin(u')\sin(v') \end{cases}.$$

If $u \neq \frac{\pi}{2} + n\pi$ or $v \neq \frac{\pi}{2} + n\pi$, the first case therefore implies:

$$\begin{cases} \cos(u) = \cos(u') \\ \cos(v) = \cos(v') \\ \sin(u)\sin(v) = \sin(u')\sin(v') \end{cases}. \tag{6.12}$$

Since the equality between cosines implies the equality between sines up to the sign, this leaves us with only two possibilities: either $\sin(u) = \sin(u')$ and $\sin(v) = \sin(v')$ or $\sin(u) = -\sin(u')$ and $\sin(v) = -\sin(v')$. Overall:

$$\begin{pmatrix} \cos(u') \\ \sin(u') \\ \cos(v') \\ \sin(v') \end{pmatrix} = \begin{pmatrix} \cos(u) \\ \sin(u) \\ \cos(v) \\ \sin(v) \end{pmatrix} \text{ or } \begin{pmatrix} \cos(u) \\ -\sin(u) \\ \cos(v) \\ -\sin(v) \end{pmatrix}$$

which in the end mean that:

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = R(0) \begin{pmatrix} u+n_u \\ v+n_v \end{pmatrix} \text{ or } R(\pi) \begin{pmatrix} u+n_u \\ v+n_v \end{pmatrix}.$$

Note that in the case where $u = \frac{\pi}{2} + n\pi$ and $v = \frac{\pi}{2} + n\pi$ the result still holds.

Case 2. Application of Equations 6.11 and substitution leads to:

$$\begin{cases} \cos(u) = \cos(u') \\ \cos(v) = \cos(v') \\ \cos(u)\cos(2v) - 2\cos(v)\sin(u)\sin(v) = \cos(2u)\cos(v) + 2\cos(u)\sin(u')\sin(v') \\ \cos(2u)\cos(v) + 2\cos(u)\sin(u)\sin(v) = \cos(u)\cos(2v) - 2\cos(v)\sin(u')\sin(v') \end{cases}$$

from which we can take the sum of the last two equations and substitute in the first two to get:

$$[\cos(u) - \cos(v)]\sin(u)\sin(v) = [\cos(u) - \cos(v)]\sin(u')\sin(v').$$

In the case where $\cos(u) - \cos(v) \neq 0$, we can simplify again, retrieve exactly system (6.12) of Case 1 and conclude similarly. Otherwise, notice that is means that $\cos(u) = \cos(v) = \cos(u') = \cos(v')$ thus $\sin(u)\sin(v) = \sin(u')\sin(v')$ and the result still holds.

Case 3. Just like Case 2, we can apply Equation (6.11), substitute and take the sum and differences of two equations to get:

$$[\cos(u) - \cos(v)]\sin(u)\sin(v) = -[\cos(u) - \cos(v)]\sin(u')\sin(v').$$

If we suppose that $\cos(u) - \cos(v) \neq 0$ (the equality case being similar to previously), then we are left with the system:

$$\begin{cases} \cos(u) = \cos(v') \\ \cos(v) = \cos(u') \\ \sin(u)\sin(v) = -\sin(u')\sin(v') \end{cases}. \tag{6.13}$$

Overall, this means that:

$$\begin{pmatrix} \cos(u') \\ \sin(u') \\ \cos(v') \\ \sin(v') \end{pmatrix} = \begin{pmatrix} \cos(u) \\ -\sin(u) \\ \cos(v) \\ \sin(v) \end{pmatrix} \text{ or } \begin{pmatrix} \cos(u) \\ \sin(u) \\ \cos(v) \\ -\sin(v) \end{pmatrix}$$

which in the end implies:

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = R(\frac{\pi}{2}) \begin{pmatrix} u + n_u \\ v + n_v \end{pmatrix} \text{ or } R(\frac{3\pi}{2}) \begin{pmatrix} u + n_u \\ v + n_v \end{pmatrix}.$$

Case 4. Finally, the proof for Case 4 combines steps from Case 1 and Case 3. Developing the last two equations leads to:

$$\begin{cases} \cos(u) = \cos(v') \\ \cos(v) = \cos(u') \\ \cos(v)\sin(u)\sin(v) = -\cos(v)\sin(u')\sin(v') \\ \cos(u)\sin(u)\sin(v) = -\cos(u)\sin(u')\sin(v') \end{cases}.$$

If $u \neq \frac{\pi}{2} + n\pi$ or $v \neq \frac{\pi}{2} + n\pi$, we can therefore simplify the full system to the same as Equation (6.13) of Case 3 and conclude similarly.

Overall, whatever the case among the four possibilities, we have shown that (u, v) and (u', v') correspond to each other up to a rotation of angle $k\frac{\pi}{2}$ and an integer translation.

□

6.3.3 Oscillator Design

Now that we have determined a relationship between uv-coordinates to satisfy the grid-preserving constraint, we focus on the design of an oscillator function. Just like the θ function of PGP (Equation (6.2)), such a function should exhibit two properties: 1) allowing expressing coefficients from Equation (6.8) as a linear combination of its components and 2) transforming translations in parameter space as a linear transformation on oscillators that only depends on the translation's uv-coordinates.

As the coefficients we are interested in in Equation (6.8) involve eight different "frequencies" in terms of u and v, namely u, $2u$, v, $2v$, $u+2v$, $2u+4v$, $2u-v$ and $4u-2v$, the straightforward approach is to consider an oscillator function ψ that contains sines and cosines of all of these frequencies:

$$\psi(u,v) := \begin{Bmatrix} \cos(u) & \cos(v) & \cos(\ u+2v) & \cos(2u-\ v) \\ \sin(u) & \sin(v) & \sin(\ u+2v) & \sin(2u-\ v) \\ \cos(2u) & \cos(2v) & \cos(2u+4v) & \cos(4u-2v) \\ \sin(2u) & \sin(2v) & \sin(2u+4v) & \sin(4u-2v) \end{Bmatrix} \in \mathbb{R}^{16}. \tag{6.14}$$

This definition for ψ allows expressing the coefficients as a linear transformation of its coordinates:

$$\begin{pmatrix} 4\cos(u) + 4\cos(v) \\ \cos(2u) + \cos(2v) \\ 4\cos(u+2v) + 4\cos(2u-v) \\ \cos(2u+4v) + \cos(4u-2v) \end{pmatrix} = \begin{pmatrix} 4\psi[0] & + & 4\psi[4] \\ \psi[2] & + & \psi[6] \\ 4\psi[8] & + & 4\psi[12] \\ \psi[10] & + & \psi[14] \end{pmatrix} := A\psi(u,v). \tag{6.15}$$

Here, we add a weighting term on some coefficients to account for their multiplying constant in the Fourier decomposition of Section 6.3.2. In the end, for two oscillators ψ and ψ' defined at two corresponding points across an edge, the pendant of Equation (6.3) of the original PGP algorithm becomes:

$$A(\psi' - \psi) = 0 \tag{6.16}$$

Additionally, since our new oscillators are only trigonometric functions of the uv-coordinates, we can perform the same reasoning as the original PGP and end up with the pendant of Equation (6.4):

$$\psi(u_j, v_j) = \hat{R}(e_u, e_v)\psi(u_i, v_i) \tag{6.17}$$

where $\hat{R}(e_u, e_v)$ is a 16×16 block diagonal matrix made of 2×2 rotation matrices whose angles are linear combinations of e_u and e_v corresponding to the frequencies of the oscillator function:

$$\hat{R}(e_u, e_v) := \begin{pmatrix} R(e_u) & 0 & & & & & & & \\ 0 & R(2e_u) & 0 & & & & & 0 & \\ & 0 & R(e_v) & 0 & & & & & \\ & & 0 & R(2e_v) & 0 & & & & \\ & & & 0 & R(e_u+2e_v) & 0 & & & \\ & & & & 0 & R(2e_u+4e_v) & 0 & & \\ & 0 & & & & 0 & R(2e_u-e_v) & 0 & \\ & & & & & & 0 & R(4e_u-2e_v) & \end{pmatrix}.$$

6.4 Parametrization Setup

Given Equations (6.16) and (6.17), we are now ready to set up an optimization problem for computing a grid-preserving parametrization of a given triangle mesh.

6.4.1 Setup per Triangle

We consider as an input triangle meshes $M = (V, E, T)$ with a valid set of feature edges and feature corners. Without loss of generality, we will assume that each triangle is adjacent to at most one feature edge and one corner (otherwise, this issue can always be solved via subdivision).

Our parametrization will be represented as a 2×2 Jacobian matrix J_t per triangle t, mapping the original triangle's coordinates in an arbitrary basis to its uv-coordinates. Using this setup, we are assured that no triangle can be broken in the final parametrization.

For each triangle $t = (i, j, k)$, we define an arbitrary basis X, Y of its tangent plane as well as a reference point r_t. We then define the coordinate function $s_t : \mathbb{R}^2 \to \mathbb{R}^2$ over the triangle as:

$$s_t(p) := p - r_t$$

. This linear function can be stored as the three values it takes at corners, namely $s_t(i), s_t(j)$ and $s_t(k)$, and extended to any point p inside the triangle using barycentric coordinates.

Additionally, we associate an oscillator function $\psi_t = \psi(r_t)$ defined at the reference point. By doing this, we are able to express the oscillator $\psi(p)$ at each point of the triangle using Equation (6.17) as:

$$\psi(p) = \hat{R}(J_t s_t(p))\psi_t.$$

This ensures oscillators coherence over each triangle chart. This setup is illustrated in Figure 6.4.

As shown in Figure 6.5, the reference point r_t of a regular triangle is taken as its center of mass. For triangles containing a feature edge with no corners, the reference point is taken as the middle of the feature. For a triangle with a feature corner, the reference point is the corner. Doing this will allow for simple expression of constraints in optimization.

Now that our variables have been defined, let us focus on the constraints to optimize.

6.4.2 Feature Constraints

For a triangle t of a feature corner, we know that its reference point r_t corresponds to the position of the corner, and that position should be on the integer grid. Since both u and v are integers, this means that all cosines of the oscillators are equal to 1 and all sines to 0. In optimization, we therefore enforce:

$$\psi_t = (1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0)^T. \qquad (\mathcal{F}_c)$$

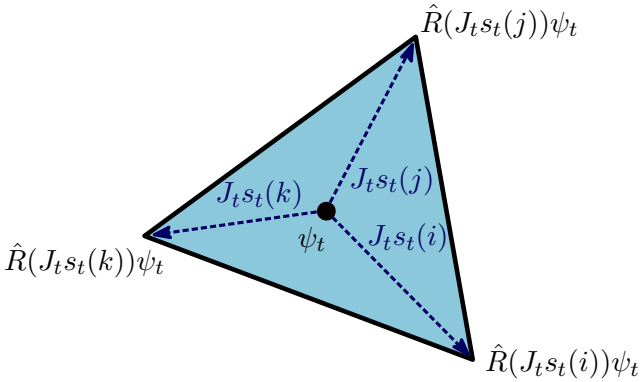


Figure 6.4: Variables setup for our problem. Given the Jacobian matrix J_t and three reference vectors $s_t(i)$, $s_t(j)$ and $s_t(k)$, we are able to express the oscillator function everywhere onto the triangle from its reference oscillator ψ_t.

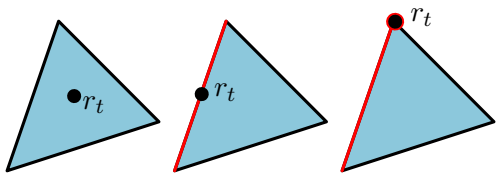


Figure 6.5: Reference point r_t for a regular triangle t is taken as its center of mass (left). For triangles near a feature edge, r_t is taken as the middle of the edge (middle). If the triangle is adjacent to a feature corner, the reference point is the corner (right).

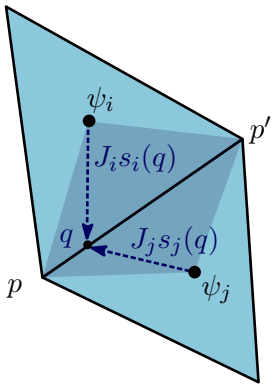
For a triangle t near a feature edge but not a corner, we placed the reference point on the edge. This means that either its u or v coordinate is an integer. We therefore enforce either:

$$\psi_t[0..3] = (1\,0\,1\,0)^T \quad \text{or} \quad \psi_t[4..7] = (1\,0\,1\,0)^T\,. \tag{$\mathcal{F}_e$}$$

The choice between u or v can be determined at initialization. When no feature edge is present on the mesh, we block one arbitrary oscillator to be of the form given by Equation ($\mathcal{F}_c$), which amounts at fixing a global translation of the uv-coordinates. As ψ will be taken as any vector of $\mathbb{R}^{16}$, these constraints prevent the optimization from converging to the trivial zero solution.

6.4.3 Adjacency Constraint

For each pair of adjacent triangles t_i and t_j, separated by an edge (p, p'), we require that the oscillators obtained by Equation (6.17) from t_i and t_j at the same point should satisfy Equation (6.16). However, it does not suffice that this condition holds at p and p', but also for every point q on the edge to guarantee coherence of the transition function (see inset Figure). We express this condition as an integral over the edge:



$$\int_p^{p'} ||A\left(\hat{R}(J_i s_i(q))\psi_i - \hat{R}(J_j s_j(q))\psi_j\right)||^2 dq = 0.$$

Since this integral is hard to optimize upon, we approximate it using the method of rectangles and instead consider points $(q_k)_{1\leqslant k\leqslant n}$ regularly sampled along the edge and satisfy:

$$\sum_{k=1}^{n} ||A\left(\hat{R}(J_i s_i(q_k))\psi_i - \hat{R}(J_j s_j(q_k))\psi_j\right)||^2 = 0.$$

We define our final energy $\mathcal{E}_{adj}$ as this quantity summed for all pairs of adjacent triangles in the mesh:

$$\mathcal{E}_{adj} := \sum_{t_i \sim t_j} \sum_{k} ||A \left(\hat{R}(J_i s_i(q_k))\psi_i - \hat{R}(J_j s_j(q_k))\psi_j \right) ||^2.$$

and we are interested in computing:

$$\begin{array}{rl} \min\limits_{J,\psi} & \mathcal{E}_{adj} \\ \text{subject to:} & (\mathcal{F}_e) \text{ and } (\mathcal{F}_c) \end{array} \tag{6.18}$$

under some constraints of injectivity, distortion and validity of oscillators.

6.4.4 Injectivity and Distortion

Problem (6.18) depends linearly on the oscillator variables and non-linearly on the Jacobian matrices. Just like in the algorithm described in Chapter 5, we need to ensure injectivity of the final parametrization, which boils down to force Jacobian matrices to have a positive determinant. This constraint is enforced using a barrier function in the exact same way as in the method of moving frames. We refer to Section 4.8.1 and 5.1.3 for more details.

Similarly, the setup of having one 2×2 matrix per triangle allows optimizing for any smooth distortion energy $\mathcal{D}$ along Problem (6.18), just as the previous chapters. We refer to Section 5.2.3 for the definition of the most common distortion functions.

6.4.5 Oscillators Validity

In the original PGP, finding a zero of the energy function meant that oscillators variables, only defined as vectors of $\mathbb{R}^4$, would have the correct form, meaning that there exists two underlying functions u, v such that $\theta = \theta(u, v)$. This is mostly because both the seam constraint (Equation (6.3)) and the oscillation equation (Equation (6.4)) involve all the components of the $\mathbb{R}^4$ vector. In our case, the adjacency constraint of Equation (6.16) only involves 8 out of the 16 coordinates of ψ. As a consequence, the set of possible vectors of $\mathbb{R}^{16}$ minimizing Problem (6.18) is larger than the set of oscillators $\psi(u, v)$ defined by Equation (6.14), even with the linear constraints on features introduced in the last section.

To alleviate this problem, we propose a regularization energy $\mathcal{E}_{valid}$ which enforces oscillators to be sine and cosine functions of u, v. Indeed, notice that a vector $\psi \in \mathbb{R}^{16}$ is of the form given by Equation (6.14) if and only if the following set of quadratic equations is satisfied:

$$\begin{cases} \psi[2i]^2 + \psi[2i+1]^2 - 1 = 0, \quad 0 \leqslant i \leqslant 8 \\ \psi[4j+2] - \psi[4j]^2 + \psi[4j+1]^2 = 0, \quad 0 \leqslant j \leqslant 4 \\ \psi[4j+3] - 2\psi[4j]\psi[4j+1] = 0, \quad 0 \leqslant j \leqslant 4 \\ \psi[8] - \psi[0]\psi[6] + \psi[1]\psi[7] = 0 \\ \psi[9] - \psi[0]\psi[7] - \psi[1]\psi[6] = 0 \\ \psi[12] - \psi[2]\psi[4] - \psi[3]\psi[5] = 0 \\ \psi[13] - \psi[2]\psi[5] + \psi[3]\psi[4] = 0 \end{cases} . \tag{$\mathcal{E}_{valid}$}$$

These equations simply constraint the norm of each $(\cos, \sin)$ pair to be 1 and utilizes trigonometric identities to make every pairs coherent with one another. Given this system, our energy

$\mathcal{E}_{valid}$ is the squared-norm of the vector of these 20 equations per oscillator for all oscillators defined on the mesh.

6.4.6 Final optimization problem

In summary, we setup an optimization problem over our variables J_t, ψ_t per triangle t in order to find a zero of $\mathcal{E}_{adj}$ and $\mathcal{E}_{valid}$. In practice, we optimize for the sum of both energies using a Ginzburg-Landau approach [Beaufort et al. 2017; Blanchi et al. 2021]. Energy $\mathcal{E}_{valid}$ is multiplied by a weighting term $\varepsilon > 0$ that will increase during optimization.

Additionally, it is possible to also optimize for some distortion $\mathcal{D}$ over Jacobians, using a weighting term λ. Overall, the full optimization problem is stated as follows:

$$\begin{aligned} \min_{J,\psi} \quad & \mathcal{E}_{adj}(J,\psi) + \varepsilon \mathcal{E}_{valid}(\psi) + \lambda \mathcal{D}(J) \\ \text{subject to:} \quad & (\mathcal{F}_e) \text{ and } (\mathcal{F}_c) \\ & \det(J_t) > 0 \end{aligned} \quad . \tag{6.19}$$

6.5 Conclusion

As mentioned in the introduction, the work presented in this chapter is still in progress. Assessing the viability of our approach on a variety of triangle meshes in a practical implementation of Problem (6.19) is an ongoing work. We would also like to demonstrate different behaviors in terms of final results depending on the distortion energy used, both in terms of final distortion and number of singularities.

7

Conclusion and Perspectives

In this final chapter, we recall our contributions and discuss their limitations as well as what remains to be achieved in the domain of quadmeshing. We will also conclude with a perspective towards the more challenging yet similar problem of hexmeshing via volume parametrization.

7.1 Summary of Contributions

This work focused on improving the classical parametrization-based quadmeshing approach, as described in Section 2.4.2. In total, we designed three different optimization algorithms. The first one turned out to be numerically impractical, the second one was declined into two versions and demonstrated state-of-the-art results, while the last one is still undergoing active research.

Cone distribution over a 4-covering In Chapter 3, we first attempted to design a continuous optimization able to automatically place quantized cones with angle defect multiple of $\pi/2$ onto a triangle mesh. The idea was to rely on a decomposition of the mesh into triangle rings and a transformation of said rings, the so-called 4-covering. We then showed that computing a flat parametrization of this new domain would imply the computation of a seamless parametrization on the original mesh. By defining and optimizing a (non-linear, non-convex) flattening energy, inspired by early conformal parametrization algorithms, we attempted at flattening the 4-covering. This approach, however, turned out to be unsuccessful due to numerical difficulties.

The method of moving frames for surface parametrization In Chapters 4 and 5, we turned our attention to Elie Cartan's method of moving frames, which we expanded to singular frame fields. Based on this theoretical framework, we derived an optimization problem that computes simultaneously a frame field and an associated parametrization, will full control over both the singularity distribution and the mapping's distortion. We showed that finding a solution of our optimization problem is equivalent to computing a rotationally seamless parametrization. We are therefore able to solve the first two steps of the usual quadmeshing pipeline in one go.

The moving frame surface parametrization algorithm turned out to be a very versatile approach to the seamless parametrization problem, as many different constraints can be optimized alongside validity of the result: distortion minimization, fixed cone positions, feature adaptation or alignment with curvature directions.

We implemented two different versions of the moving frame method: one where surface charts lie around vertices (Chapter 4) and one where they correspond to the faces of the input mesh (Chapter 5). Only the former is guaranteed to not form double-coverings and therefore output

a locally-injective parametrization. On the other hand, the formulation of the latter is simpler, its implementation shorter and it exhibits better performance in practice.

Grid-preserving parametrization in a single optimization Finally, a significant part of our research work has been dedicated to generalizing the method of moving frames, both for quadmeshing and also for hexmeshing (see Section 7.3). As far as surfaces are concerned, the most promising approach is an attempt at merging it with the quantization step of the quadmeshing pipeline, allowing us to compute a grid-preserving parametrization by solving a single continuous optimization problem. Taking inspiration from the *Periodic Global Parametrization* method of Ray et al. [2006], we designed a set of constraints over periodic variables such that their satisfaction is equivalent to the existence of a grid-preserving parametrization. These constraints are set up in a non-linear least-square optimization problem over a triangle mesh.

Overall, our efforts have been focused on proposing formulations of the seamless parametrization problem that came from the continuous mathematical case and did not involve mixed-integer optimization nor greedy algorithms. Retrospectively, one common aspect of all the methods described in this work is the optimization of an already known system (LSCM for Chapter 3, Cartan's structure equations in Chapter 4 and 5, the PGP least-square system in Chapter 6) where we allowed previously constant quantities to change. This placed us in a tradeoff between two worlds that we argue was worth exploring. On the one hand, optimizing along other variables broke some properties of the original system, notably it's linearity. This forced us to use non-linear optimization algorithm like Levenberg-Marquardt or L-BFGS to solve the "harder" problem, hence an increase in computational cost and a loss in convergence guarantees. On the other hand, those newly defined energies allowed us to find new solutions to the problem of parametrization that were out of reach of previous approaches.

7.2 Future Works in Quadmeshing

While many advances have been performed in the last decades regarding the problem of quadmeshing – both with direct and parametrization-based approaches – the ultimate quadmeshing algorithm is still out of reach. We list here a few remaining issues to be fixed and improvements to be made, while giving some ideas of future works to be built upon our contributions.

Robustness, limit cycles Although they create quadmeshes with less irregular vertices than other methods, parametrization approaches to quadmeshing still do not match robustness of direct methods like front propagation [Remacle et al. 2013]. This can be partially explained via the variational nature of the problem involved: the non-convex optimization can become stuck in local minima or towards unreachable asymptotic solutions like limit cycles (Figure 4.14). In Chapters 4 and 5, we indeed report some failure cases on our dataset of models where the optimizer failed to converge to a valid solution. Moreover, the quality of the initial representation of the surface, that is the input triangle mesh, also plays a great role in terms of numerical stability.

An idea to regularize seamless parametrization optimization would be to fix lengths of boundary and feature edges to be the same in parameter space as in real space. We believe that such a constraint has the potential to fix limit cycle integrability issues like the one depicted in Figure 4.14a of Chapter 4, since matching boundaries would be forced to have different lengths. This could be easily integrated into the method of moving frames using linear constraints. As

far as grid-preserving parametrizations are concerned, we believe that an implementation of the optimization problem described in Chapter 6 would also not suffer from these issues. Yet, convergence behavior of such a method remains to be assessed.

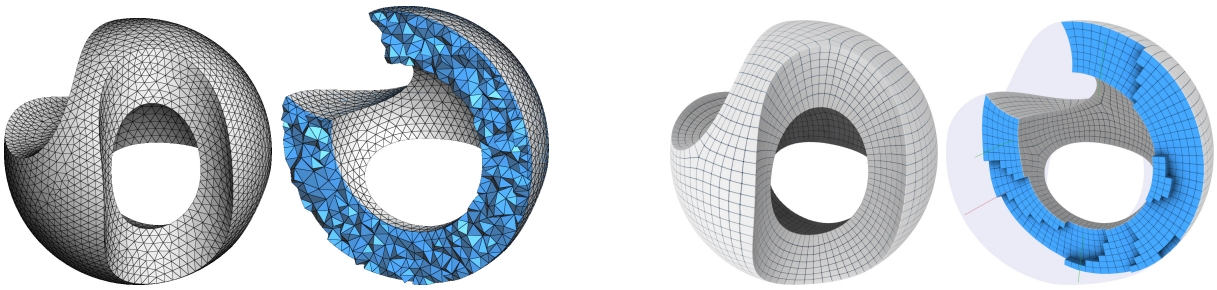
Non-uniform sizing, anisotropy Quadmeshes are often used in numerical simulations for their ability to form highly anisotropic elements while their angle stay near $\pi/2$ [Garimella and Shephard 2000; Marcum et al. 2017], for instance near interfaces where precision along the normal direction is more important than in the others. Unfortunately, such constraints are incompatible with our moving frame method, since they require a global frame to be defined for directions of anisotropy to be consistent. On the other hand, it could theoretically be integrated into our grid-preserving algorithm (Chapter 6) in the form of anisotropic deformation of the triangles.

In the simpler case where the difference in sizing is isotropic, meaning that quad elements resemble squares of different sizes over the mesh, a specific sizing constraint could be integrated in both methods as a penalty over Jacobian matrices. Given their ability to freely place singularities, we believe that our formulations would still be able to produce a valid seamless parametrization matching the desired sizing, at the expense of more compensating cones. Nonetheless, whether this kind of approaches is more efficient or interesting than an a posteriori editing of the mesh remains to be seen.

Quad sizing vs distortion In our work, we discussed several times the tradeoff between the number of singularities and distortion of a seamless parametrization. As pointed out by Myles and Zorin [2013], singularities can be added indefinitely to make the parametrization converge to a perfect isometry. Yet, this process does not create a good quality quadmesh as elements near areas with many singularities become indefinitely small. In practice, the sizing of quadrilateral elements is as important as their distortion. Thus, the true problem of parametrization for quadmeshing would consist of finding the optimal set of singularities given a fixed element size (or resolution of elements) in order for all the elements to be as close to squares (eventually rectangles in anisotropic cases) as possible.

So far, solving this problem is only indirectly done by our method since the element size is not explicitly incorporated in neither the method of moving frames, nor our grid-preserving parametrization approach.

Computational efficiency and initializations Finally, instead of solving for a grid-preserving parametrization in one step, one could instead imagine a version of the pipeline where each step still has control over the previously computed degrees of freedom and is able to correct any previously made decision. In other words, instead of computing everything from scratch, one could rely on a series of good initializations fed to increasingly complex optimization problems. Such an approach would potentially reduce the large computation times of our algorithms, all without hurting the quality of the results. For instance, given a smooth frame field, a parametrization algorithm able to add, move or delete singularities not only for integrability (as [Diamanti et al. 2015]) but for the minimization of any criterion (without creating T-junctions like Ray et al. [2006]) would be desirable. Likewise, exploiting the tradeoff between additional singularities and distortion in the quantization step (like our attempt in Chapter 6) could be an interesting future work.



(a) (b)

Figure 7.1: Examples of tetrahedral and hexahedral meshes of the *sculpt* model, rendered as full objects as well as sliced along the z axis. (a) Tetmesh rendered in *Graphite* 3[10]. (b) Hexmesh provided by Corman and Crane [2019] and rendered in *Hexalab* [Bracci et al. 2019].

7.3 Towards Parametrization Methods for Hexmeshing

While we focused so far on the case of surfaces and quadmeshes, a related and challenging problem in the field of geometry processing is the automatic generation of hexahedral meshes. In numerical simulations, tetrahedral and hexahedral meshes are volume meshes made of vertices, edges, faces and *cells*, the latter being tetrahedra (4 triangular faces) or hexahedra (6 quadrangular faces) respectively (Figure 7.1). They are a direct generalization of triangular and quadrangular meshes for the representation of volumetric data. While the generation of tetmeshes is mostly understood and mastered [Geuzaine and Remacle 2009; Si 2015], the problem of generating hexmeshes still poses theoretical and practical challenges.

To conclude this manuscript, we give a quick overview of the specific challenges posed by the hexmeshing problem, and discuss how our work can be extended to this three-dimensional case.

7.3.1 Hexmeshing

Automatically generating a hexmesh from a tetmesh is a well-studied problem in geometry processing [Pietroni et al. 2023]. Apart from direct approaches like front propagation or tree-based meshing, which have been successfully applied to this new problem, the parametrization approach has also been investigated. However, this generalization did not come without its own set of challenges and robustness issues. To cite a few, we can observe for instance that conformal maps are not expressive enough in volume (see Liouville's theorem on conformal maps), or that Morse-Smale complexes ni volumes are not a hexahedral decomposition [Ling et al. 2014].

Yet, the most important distinction between the volume and surface case is perhaps the behavior of frame fields. As they represent a set of orthogonal directions on a manifold, frame fields can be generalized in 3D in a straightforward way, that is by assigning a set of six directions $\{u, -u, v, -v, w, -w\}$ to each element of a mesh. Given a suitable representation of this set, like the spherical harmonics decomposition of Ray et al. [2016], the Laplacian-based smoothing algorithm remains the same. However, singularities of the field do not appear at isolated points anymore, but rather form a network of lines called a *singularity graph* which ultimately correspond to edges in the hexmesh incident to 3 or 5 hexes instead of 4. This graph, depicted in Figure 7.2 on a few examples, is therefore critical to the quality of the final hexmesh.

Unfortunately, smooth frame fields in 3D tend to exhibit very noisy singularity graphs and

[10] `https://github.com/BrunoLevy/GraphiteThree`

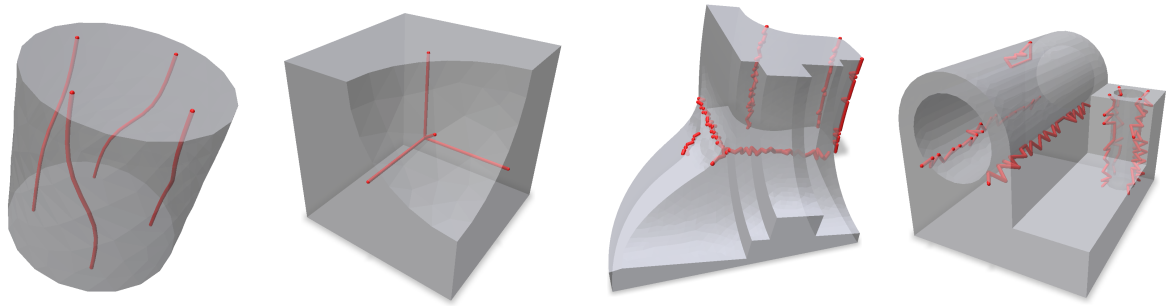


Figure 7.2: In the 3D case, singularities of a cross field form a set of lines traversing the volume called a singularity graph. Here, the singularity graph of the four shapes is a subset of edges of the mesh, depicted in red. The two leftmost examples are hand made to be valid graphs. The other two examples were generated using the frame field smoothing algorithm of Ray et al. [2016].

often unmeshable configurations (like on the rightmost example of Figure 7.2). Unlike the surface case where theoretical results have been proved on the validity of a given singularity distribution [Shen et al. 2022], no such result exists for singularity graphs. The Gauss-Bonnet theorem, which gives in 2D a simple criterion for the validity of a cone distribution (let aside some degenerated cases like limit cycles), does not hold in higher dimensions. Likewise, no criterion exists to this day that can determine if a given frame field is integrable into a seamless parametrization. Some very recent works have been made in this direction [Liu and Bommes 2023] but are only able to fix local meshability issues.

If a meshable frame field is provided, a grid-preserving parametrization can be recovered using the *CubeCover* method of Nieser et al. [2011], which is a direct extension of the *QuadCover* algorithm for surfaces [Kälberer et al. 2007]. Alternatively, if a valid singularity graph is provided on a tetrahedral mesh, an associated frame field and parametrization can also be computed using the method of Liu et al. [2018b] or Corman and Crane [2019]. The quantization step has also been analyzed: Brückler et al. [2022] formulate it as a mixed-integer problem over the 3D equivalent of a T-mesh. Finally, hexahedral elements can be extracted reliably using the algorithm from Lyon et al. [2016]. Overall, only the problem of findings a valid singularity graph remains truly open.

7.3.2 The Method of Moving Frames for Volume Parametrization

Designing a parametrization algorithm that can theoretically be applied to both the surface and the volume case is challenging as most approaches for surfaces do not generalize well in 3D. Fortunately, this is not the case of the method of moving frames, as Cartan's structure equations work for smooth manifolds in any finite dimension.

This observation is a key motivation for our algorithms of Chapters 4 and 5, which can be applied to 3D with minimal changes in the formulation. By using the same representation of frames as Ray et al. [2016] and splitting each input tetrahedra into four hexahedral charts centered around vertices (as in Figure 7.3), we can discretize the structure equations as a non-linear least-square problem similar to Problem (4.11). Although a cell-based discretization is also possible (just like the face-based alternative discretization discussed in Chapter 5), it would force singular lines to stay on the edges of the base mesh. On the other hand, the vertex-based approach allows singularities to move freely inside the domain. We thus expect the optimization to be smoother in this case.

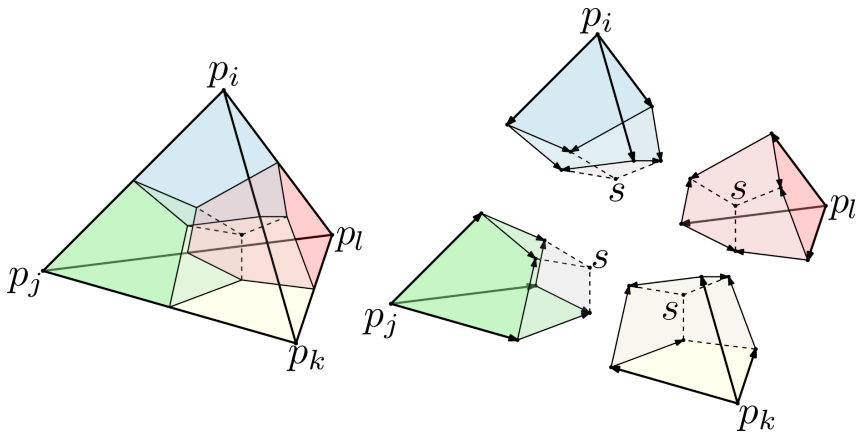


Figure 7.3: The moving frame method described in Chapter 4 can be generalized to volume meshes by considering splitting each tetrahedron into four hexahedra by adding central points to each cell and face. In this setup, singularities can appear as lines linking centers of faces to centers of tetrahedra.

In the absence of any tractable criterion for validity of a singularity graph, our intuition for this approach lies in the tautological fact that a frame field is integrable if it can be integrated. As such, we expect the simultaneous optimization of a frame field and an associated parametrization in a moving frame fashion to be more successful than just the smoothing of the former, the parametrization here acting as a regularization forcing the frame field to stay integrable during optimization. This would result in the first algorithm for computing a valid singularity graph and its associated frame field and seamless parametrization all at once.

An implementation of this algorithm has been attempted during this thesis but results still need to be improved. In particular, the optimization often got stuck in saddle points where the frame field is smooth but its singularity graph is invalid, leading to highly distorted elements. While the idea of using the moving frame method to automatically compute valid singularity graphs seems promising, additional work is therefore required to make it work in practice.

Bibliography

Noam Aigerman, Kunal Gupta, Vladimir G. Kim, Siddhartha Chaudhuri, Jun Saito, and Thibault Groueix. 2022. Neural Jacobian Fields: Learning Intrinsic Mappings of Arbitrary Meshes. arXiv:2205.02904 [cs]

Noam Aigerman and Yaron Lipman. 2015. Orbifold Tutte Embeddings. *ACM Transactions on Graphics* 34, 6 (Nov. 2015), 190:1–190:12. https://doi.org/10.1145/2816795.2818099

Noam Aigerman and Yaron Lipman. 2016. Hyperbolic Orbifold Tutte Embeddings. *ACM Transactions on Graphics* 35, 6 (Nov. 2016), 1–14. https://doi.org/10.1145/2980179.2982412

Christie Alappat, Achim Basermann, Alan R. Bishop, Holger Fehske, Georg Hager, Olaf Schenk, Jonas Thies, and Gerhard Wellein. 2020. A Recursive Algebraic Coloring Technique for Hardware-Efficient Symmetric Sparse Matrix-Vector Multiplication. *ACM Transactions on Parallel Computing* 7, 3, Article 19 (June 2020). https://doi.org/10.1145/3399732

Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Levy, and Mathieu Desbrun. 2003. Anisotropic Polygonal Remeshing. (April 2003).

Pierre Alliez, Giuliana Ucelli, Craig Gotsman, and Marco Attene. 2008. Recent Advances in Remeshing of Surfaces. In *Shape Analysis and Structuring*, Gerald Farin, Hans-Christian Hege, David Hoffman, Christopher R. Johnson, Konrad Polthier, Martin Rumpf, Leila De Floriani, and Michela Spagnuolo (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 53–82. https://doi.org/10.1007/978-3-540-33265-7_2

Douglas N. Arnold, Daniele Boffi, and Richard S. Falk. 2002. Approximation by Quadrilateral Finite Elements. *Math. Comp.* 71, 239 (March 2002), 909–922. https://doi.org/10.1090/S0025-5718-02-01439-4

Thierry Aubin. 1998. *Some Nonlinear Problems in Riemannian Geometry.* Springer Science & Business Media.

Thomas F Banchoff. 1970. Critical points and curvature for embedded polyhedral surfaces. *The American Mathematical Monthly* 77, 5 (1970), 475–485.

Pierre-Alexandre Beaufort, Jonathan Lambrechts, François Henrotte, Christophe Geuzaine, and Jean-François Remacle. 2017. Computing Cross Fields A PDE Approach Based on the Ginzburg-Landau Theory. *Procedia Engineering* 203 (2017), 219–231. https://doi.org/10.1016/j.proeng.2017.09.799

Mirela Ben-Chen, Craig Gotsman, and Guy Bunin. 2008. Conformal Flattening by Curvature Prescription and Metric Scaling. *Computer Graphics Forum* 27, 2 (2008), 449–458. https://doi.org/10.1111/j.1467-8659.2008.01142.x

Ted D. Blacker and Michael B. Stephenson. 1991. Paving: A New Approach to Automated Quadrilateral Mesh Generation. *Internat. J. Numer. Methods Engrg.* 32, 4 (1991), 811–847. https://doi.org/10.1002/nme.1620320410

Victor Blanchi, Étienne Corman, Nicolas Ray, and Dmitry Sokolov. 2021. Global Parametrization Based on Ginzburg-Landau Functional. In *Numerical Geometry, Grid Generation and Scientific Computing*, Vladimir A. Garanzha, Lennard Kamenski, and Hang Si (Eds.). Vol. 143. Springer International Publishing, Cham, 251–262. https://doi.org/10.1007/978-3-030-76798-3_16

Matthias Bollhöfer, Aryan Eftekhari, Simon Scheidegger, and Olaf Schenk. 2019. Large-Scale Sparse Inverse Covariance Matrix Estimation. *SIAM Journal on Scientific Computing* 41, 1 (2019), A380–A401. https://doi.org/10.1137/17M1147615 arXiv:https://doi.org/10.1137/17M1147615

Matthias Bollhöfer, Olaf Schenk, Radim Janalik, Steve Hamm, and Kiran Gullapalli. 2020. State-of-the-Art Sparse Direct Solvers. *Parallel algorithms in computational science and engineering* (2020), 3–33. https://doi.org/10.1007/978-3-030-43736-7_1

David Bommes, Marcel Campen, Hans-Christian Ebke, Pierre Alliez, and Leif Kobbelt. 2013a. Integer-Grid Maps for Reliable Quad Meshing. *ACM Transactions on Graphics* 32, 4 (July 2013), 98:1–98:12. https://doi.org/10.1145/2461912.2462014

David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Claudio Silva, Marco Tarini, and Denis Zorin. 2013b. Quad-Mesh Generation and Processing: A Survey. *Computer Graphics Forum* 32, 6 (2013), 51–76. https://doi.org/10.1111/cgf.12014

David Bommes, Henrik Zimmer, and Leif Kobbelt. 2009. Mixed-Integer Quadrangulation. *ACM Transactions on Graphics* 28, 3 (July 2009), 1–10. https://doi.org/10.1145/1531326.1531383

Houman Borouchaki and Pascal J. Frey. 1998. Adaptive Triangular–Quadrilateral Mesh Generation. *Internat. J. Numer. Methods Engrg.* 41, 5 (1998), 915–934. https://doi.org/10.1002/(SICI)1097-0207(19980315)41:5<915::AID-NME318>3.0.CO;2-Y

Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Levy. 2010. *Polygon Mesh Processing*. CRC Press.

Matteo Bracci, Marco Tarini, Nico Pietroni, Marco Livesu, and Paolo Cignoni. 2019. HexaLab.net: An online viewer for hexahedral meshes. *Computer-Aided Design* 110 (2019), 24–36.

Hendrik Brückler, David Bommes, and Marcel Campen. 2022. Volume Parametrization Quantization for Hexahedral Meshing. *ACM Transactions on Graphics* 41, 4 (July 2022), 1–19. https://doi.org/10.1145/3528223.3530123

Marcel Campen, David Bommes, and Leif Kobbelt. 2015. Quantized Global Parametrization. *ACM Transactions on Graphics* 34, 6 (Nov. 2015), 1–12. https://doi.org/10.1145/2816795.2818140

Marcel Campen, Hanxiao Shen, Jiaran Zhou, and Denis Zorin. 2019. Seamless Parametrization with Arbitrary Cones for Arbitrary Genus. *ACM Transactions on Graphics* 39, 1 (Dec. 2019), 2:1–2:19. https://doi.org/10.1145/3360511

Élie Cartan et al. 2001. *Riemannian Geometry in an Orthogonal Frame: From Lectures Delivered by Lie Cartan at the Sorbonne in 1926-1927*. World Scientific.

E Catmull and J Clark. 1998. Recursively Generated 8-Spline Surfaces on Arbitrary Topological Meshes. (July 1998).

Edward Chien, Zohar Levi, and Ofir Weber. 2016. Bounded Distortion Parametrization in the Space of Metrics. *ACM Transactions on Graphics* 35, 6 (Nov. 2016), 1–16. https://doi.org/10.1145/2980179.2982426

P.G. Ciarlet and P.-A. Raviart. 1972. Interpolation Theory over Curved Elements, with Applications to Finite Element Methods. *Computer Methods in Applied Mechanics and Engineering* 1, 2 (Aug. 1972), 217–249. https://doi.org/10.1016/0045-7825(72)90006-0

S. Claici, M. Bessmeltsev, S. Schaefer, and J. Solomon. 2017. Isometry-Aware Preconditioning for Mesh Parameterization. *Computer Graphics Forum* 36, 5 (2017), 37–47. https://doi.org/10.1111/cgf.13243

David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. 2004. Variational Shape Approximation. (2004).

David Cohen-Steiner and Jean-Marie Morvan. 2003. Restricted Delaunay Triangulations and Normal Cycle. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry (SCG '03)*. Association for Computing Machinery, New York, NY, USA, 312–321. https://doi.org/10.1145/777792.777839

Etienne Corman and Keenan Crane. 2019. Symmetric Moving Frames. *ACM Transactions on Graphics* 38, 4 (Aug. 2019), 1–16. https://doi.org/10.1145/3306346.3323029

Yoann Coudert-Osmont, David Desobry, Martin Heistermann, David Bommes, Nicolas Ray, and Dmitry Sokolov. 2023. Quad Mesh Quantization Without a T-Mesh. *Computer Graphics Forum* n/a, n/a (2023), e14928. https://doi.org/10.1111/cgf.14928

Keenan Crane. 2018. Discrete Differential Geometry: An Applied Introduction. (2018).

Keenan Crane, Mathieu Desbrun, and Peter Schröder. 2010. Trivial Connections on Discrete Surfaces. *Computer Graphics Forum* 29 (July 2010), 1525–1533. https://doi.org/10.1111/j.1467-8659.2010.01761.x

Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Robust fairing via conformal curvature flow. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–10.

Keenan Crane and Max Wardetzky. 2017. A Glimpse into Discrete Differential Geometry. *Notices of the American Mathematical Society* 64, 10 (Nov. 2017), 1153–1159. https://doi.org/10.1090/noti1578

George Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems* 2, 4 (1989), 303–314.

Fernando de Goes, Mathieu Desbrun, and Yiying Tong. 2016. Vector Field Processing on Triangle Meshes. In *ACM SIGGRAPH 2016 Courses (SIGGRAPH '16)*. Association for Computing Machinery, New York, NY, USA, 1–49. https://doi.org/10.1145/2897826.2927303

Henri Paul de Saint-Gervais. 2016. *Uniformization of Riemann surfaces.*

Mathieu Desbrun, Anil N. Hirani, Melvin Leok, and Jerrold E. Marsden. 2005. Discrete Exterior Calculus. arXiv:math/0508341

Mathieu Desbrun, Mark Meyer, and Pierre Alliez. 2002. Intrinsic Parameterizations of Surface Meshes. *Computer Graphics Forum* 21, 3 (2002), 209–218. https://doi.org/10.1111/1467-8659.00580

David Desobry, Yoann Coudert-Osmont, Etienne Corman, Nicolas Ray, and Dmitry Sokolov. 2021. Designing 2D and 3D Non-Orthogonal Frame Fields. *Computer-Aided Design* 139 (Oct. 2021), 103081. https://doi.org/10.1016/j.cad.2021.103081

David Desobry, François Protais, Nicolas Ray, Etienne Corman, and Dmitry Sokolov. 2022. Frame Fields for CAD Models. (Jan. 2022).

Gouri Dhatt, Emmanuel Lefrançois, and Gilbert Touzot. 2012. *Finite element method.* John Wiley & Sons.

Olga Diamanti, Amir Vaxman, Daniele Panozzo, and Olga Sorkine-Hornung. 2014. Designing N-PolyVector Fields with Complex Polynomials. *Computer Graphics Forum* 33, 5 (2014), 1–11. https://doi.org/10.1111/cgf.12426

Olga Diamanti, Amir Vaxman, Daniele Panozzo, and Olga Sorkine-Hornung. 2015. Integrable PolyVector Fields. *ACM Transactions on Graphics* 34, 4 (July 2015), 1–12. https://doi.org/10.1145/2766906

Manfredo do Carmo, P. 1992. *Riemannian Geometry* (birkhäuser ed.).

Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John Hart. 2006. Spectral Surface Quadrangulation. *ACM Trans. Graph.* 25 (July 2006), 1057–1066. https://doi.org/10.1145/1141911.1141993

Xingyi Du, Noam Aigerman, Qingnan Zhou, Shahar Z. Kovalsky, Yajie Yan, Danny M. Kaufman, and Tao Ju. 2020. Lifting Simplices to Find Injectivity. *ACM Transactions on Graphics* 39, 4 (Aug. 2020), 120:120:1–120:120:17. https://doi.org/10.1145/3386569.3392484

Hans-Christian Ebke, David Bommes, Marcel Campen, and Leif Kobbelt. 2013. QEx: Robust Quad Mesh Extraction. *ACM Transactions on Graphics* 32, 6 (Nov. 2013), 168:1–168:10. https://doi.org/10.1145/2508363.2508372

Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. 1995. Multiresolution Analysis of Arbitrary Meshes. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. Association for Computing Machinery, New York, NY, USA, 173–182. https://doi.org/10.1145/218380.218440

Leonhard Euler. 1758. Elementa doctrinae solidorum. *Novi commentarii academiae scientiarum Petropolitanae* (1758), 109–140.

Qing Fang, Wenqing Ouyang, Mo Li, Ligang Liu, and Xiao-Ming Fu. 2021. Computing Sparse Cones with Bounded Distortion for Conformal Parameterizations. *ACM Transactions on Graphics* 40, 6 (Dec. 2021), 1–9. https://doi.org/10.1145/3478513.3480526

Xianzhong Fang, Hujun Bao, Yiying Tong, Mathieu Desbrun, and Jin Huang. 2018. Quadrangulation through Morse-Parameterization Hybridization. *ACM Transactions on Graphics* 37, 4 (Aug. 2018), 1–15. https://doi.org/10.1145/3197517.3201354

Michael S. Floater. 1997. Parametrization and Smooth Approximation of Surface Triangulations. *Computer Aided Geometric Design* 14, 3 (April 1997), 231–250. https://doi.org/10.1016/S0167-8396(96)00031-3

Xiao-Ming Fu and Yang Liu. 2016. Computing Inversion-Free Mappings by Simplex Assembly. *ACM Transactions on Graphics* 35, 6 (Nov. 2016), 1–12. https://doi.org/10.1145/2980179.2980231

Xifeng Gao, Hanxiao Shen, and Daniele Panozzo. 2019. Feature Preserving Octree-Based Hexahedral Meshing. *Computer Graphics Forum* 38, 5 (2019), 135–149. https://doi.org/10.1111/cgf.13795

Vladimir Garanzha, Igor Kaporin, Liudmila Kudryavtseva, François Protais, David Desobry, and Dmitry Sokolov. 2022. Lowest Distortion Mappings and Equidistribution Principle. (Nov. 2022).

Vladimir Garanzha, Igor Kaporin, Liudmila Kudryavtseva, François Protais, Nicolas Ray, and Dmitry Sokolov. 2021. Foldover-Free Maps in 50 Lines of Code. *ACM Transactions on Graphics* 40, 4 (Aug. 2021), 1–16. https://doi.org/10.1145/3450626.3459847

Rao V. Garimella and Mark S. Shephard. 2000. Boundary Layer Mesh Generation for Viscous Flow Simulations. *Internat. J. Numer. Methods Engrg.* 49, 1-2 (2000), 193–218. https://doi.org/10.1002/1097-0207(20000910/20)49:1/2<193::AID-NME929>3.0.CO;2-R

Christophe Geuzaine and Jean-François Remacle. 2009. Gmsh: A 3-D Finite Element Mesh Generator with Built-in Pre- and Post-Processing Facilities. *Internat. J. Numer. Methods Engrg.* 79, 11 (2009), 1309–1331. https://doi.org/10.1002/nme.2579

Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. 2018. AtlasNet: A Papier-M\âch\'e Approach to Learning 3D Surface Generation. https://doi.org/10.48550/arXiv.1802.05384 arXiv:1802.05384 [cs]

Xianfeng Gu and Shing-Tung Yau. 2003. Global Conformal Surface Parameterization. *Eurographics Symposium on Geometry Processing* (Jan. 2003).

Aaron Hertzmann and Denis Zorin. 2000. Illustrating Smooth Surfaces. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '00*. ACM Press, Not Known, 517–526. https://doi.org/10.1145/344779.345074

Anil N Hirani. 2003. Discrete Exterior Calculus. (2003).

Kai Hormann and Guenther Greiner. 2000. MIPS: An Efficient Global Parametrization Method. *Curve and Surface Design: Saint-Malo* 2000 (July 2000), 10.

Yasushi Ito, Alan M. Shih, and Bharat K. Soni. 2009. Octree-Based Reasonable-Quality Hexahedral Mesh Generation Using a New Set of Refinement Templates. *Internat. J. Numer. Methods Engrg.* 77, 13 (2009), 1809–1833. https://doi.org/10.1002/nme.2470

Fabrice Jaillet and Claudio Lobos. 2022. Fast Quadtree/Octree Adaptive Meshing and Re-Meshing with Linear Mixed Elements. *Engineering with Computers* 38, 4 (Aug. 2022), 3399–3416. https://doi.org/10.1007/s00366-021-01330-w

Tengfei Jiang, Xianzhong Fang, Jin Huang, Hujun Bao, Yiying Tong, and Mathieu Desbrun. 2015. Frame Field Generation through Metric Customization. *ACM Transactions on Graphics* 34 (July 2015), 40:1–40:11. https://doi.org/10.1145/2766927

Felix Kälberer, Matthias Nieser, and Konrad Polthier. 2007. QuadCover - Surface Parameterization Using Branched Coverings. *Computer Graphics Forum* 26, 3 (2007), 375–384. https://doi.org/10.1111/j.1467-8659.2007.01060.x

Liliya Kharevych, Boris Springborn, and Peter Schröder. 2006. Discrete Conformal Mappings via Circle Patterns. *ACM Transactions on Graphics* 25, 2 (April 2006), 412–438. https://doi.org/10.1145/1138450.1138461

Andrei Khodakovsky, Nathan Litke, and Peter Schröder. 2003. Globally Smooth Parameterizations with Low Distortion. *ACM Transactions on Graphics* 22, 3 (July 2003), 350–357. https://doi.org/10.1145/882262.882275

Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Globally Optimal Direction Fields. *ACM Transactions on Graphics* 32, 4 (July 2013), 1–10. https://doi.org/10.1145/2461912.2462005

Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2015. Stripe Patterns on Surfaces. *ACM Transactions on Graphics* 34, 4 (July 2015), 1–11. https://doi.org/10.1145/2767000

Marin Kobilarov, Keenan Crane, and Mathieu Desbrun. 2009. Lie Group Integrators for Animation and Control of Vehicles. *ACM Transactions on Graphics* 28, 2 (May 2009), 16:1–16:14. https://doi.org/10.1145/1516522.1516527

Vladislav Kraevoy and Alla Sheffer. 2004. Cross-Parameterization and Compatible Remeshing of 3D Models. (Aug. 2004).

Steven Krantz and Harold Parks. 2008. *Geometric Integration Theory.* Birkhäuser Boston, Boston. https://doi.org/10.1007/978-0-8176-4679-0

Yu-Kun Lai, Miao Jin, Xuexiang Xie, Ying He, Jonathan Palacios, Eugene Zhang, Shi-Min Hu, and Xianfeng Gu. 2010. Metric-Driven RoSy Field Design and Remeshing. *IEEE Transactions on Visualization and Computer Graphics* 16, 1 (Jan. 2010), 95–108. https://doi.org/10.1109/TVCG.2009.59

Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. 2015. Numba: A LLVM-based Python JIT Compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC (LLVM '15).* Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/2833157.2833162

C.K. Lee and S.H. Lo. 1994. A New Scheme for the Generation of a Graded Quadrilateral Mesh. *Computers & Structures* 52, 5 (Sept. 1994), 847–857. https://doi.org/10.1016/0045-7949(94)90070-1

Zohar Levi. 2021. Direct Seamless Parametrization. *ACM Transactions on Graphics* 40, 1 (Feb. 2021), 1–14. https://doi.org/10.1145/3439828

Zohar Levi. 2023. Shear-Reduced Seamless Parametrization. *Computer Aided Geometric Design* 101 (April 2023), 102179. https://doi.org/10.1016/j.cagd.2023.102179

Bruno Lévy and Yang Liu. 2010. L_p Centroidal Voronoi Tessellation and Its Applications. *ACM Transactions on Graphics* 29, 4 (July 2010), 1–11. https://doi.org/10.1145/1778765.1778856

Bruno Lévy and Jean-Laurent Mallet. 1998. Non-Distorted Texture Mapping for Sheared Triangulated Meshes. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '98*. ACM Press, Not Known, 343–352. https://doi.org/10.1145/280814.280930

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. 2002. Least Squares Conformal Maps for Automatic Texture Atlas Generation. *ACM Transactions on Graphics* 21, 3 (July 2002), 362–371. https://doi.org/10.1145/566654.566590

Mo Li, Qing Fang, Wenqing Ouyang, Ligang Liu, and Xiao-Ming Fu. 2022. Computing Sparse Integer-Constrained Cones for Conformal Parameterizations. *ACM Transactions on Graphics* 41, 4 (July 2022), 58:1–58:13. https://doi.org/10.1145/3528223.3530118

Minchen Li, Danny M. Kaufman, Vladimir G. Kim, Justin Solomon, and Alla Sheffer. 2018. OptCuts: Joint Optimization of Surface Cuts and Parameterization. *ACM Transactions on Graphics* 37, 6 (Dec. 2018), 1–13. https://doi.org/10.1145/3272127.3275042

Xinghua Liang, Mohamed S. Ebeida, and Yongjie Zhang. 2010. Guaranteed-Quality All-Quadrilateral Mesh Generation with Feature Preservation. *Computer Methods in Applied Mechanics and Engineering* 199, 29-32 (June 2010), 2072–2083. https://doi.org/10.1016/j.cma.2010.03.007

Ruotian Ling, Jin Huang, Bert Jüttler, Feng Sun, Hujun Bao, and Wenping Wang. 2014. Spectral Quadrangulation with Feature Curve Alignment and Element Size Control. *ACM Transactions on Graphics* 34, 1 (Dec. 2014), 1–11. https://doi.org/10.1145/2653476

Yaron Lipman. 2012. Bounded Distortion Mapping Spaces for Triangular Meshes. *ACM Transactions on Graphics* 31, 4 (Aug. 2012), 1–13. https://doi.org/10.1145/2185520.2185604

Yaron Lipman, Daniel Cohen-Or, Ran Gal, and David Levin. 2007. Volume and Shape Preservation via Moving Frame Manipulation. *ACM Transactions on Graphics* 26, 1 (Jan. 2007), 5–es. https://doi.org/10.1145/1189762.1189767

Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen-Or. 2005. Linear Rotation-Invariant Coordinates for Meshes. *ACM Transactions on Graphics* 24, 3 (July 2005), 479–487. https://doi.org/10.1145/1073204.1073217

Dong C. Liu and Jorge Nocedal. 1989. On the Limited Memory BFGS Method for Large Scale Optimization. *Mathematical Programming* 45, 1-3 (Aug. 1989), 503–528. https://doi.org/10.1007/BF01589116

Heng Liu and David Bommes. 2023. Locally Meshable Frame Fields. 42, 4 (2023).

Heng Liu, Paul Zhang, Edward Chien, Justin Solomon, and David Bommes. 2018b. Singularity-Constrained Octahedral Fields for Hexahedral Meshing. *ACM Transactions on Graphics* 37, 4 (Aug. 2018), 1–17. https://doi.org/10.1145/3197517.3201344

Hao-Yu Liu, Zhong-Yuan Liu, Zheng-Yu Zhao, Ligang Liu, and Xiao-Ming Fu. 2020. Practical Fabrication of Discrete Chebyshev Nets. (2020). https://doi.org/10.1111/cgf.14123

Ligang Liu, Chunyang Ye, Ruiqi Ni, and Xiao-Ming Fu. 2018a. Progressive Parameterizations. *ACM Transactions on Graphics* 37, 4 (Aug. 2018), 1–12. https://doi.org/10.1145/3197517.3201331

Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. 2008. A Local/Global Approach to Mesh Parameterization. *Computer Graphics Forum* 27, 5 (2008), 1495–1504. https://doi.org/10.1111/j.1467-8659.2008.01290.x

Max Lyon, David Bommes, and Leif Kobbelt. 2016. HexEx: Robust Hexahedral Mesh Extraction. *ACM Transactions on Graphics* 35, 4 (July 2016), 1–11. https://doi.org/10.1145/2897824.2925976

M. Lyon, M. Campen, and L. Kobbelt. 2021. Quad Layouts via Constrained T-Mesh Quantization. *Computer Graphics Forum* 40, 2 (May 2021), 305–314. https://doi.org/10.1111/cgf.142634

David L. Marcum, Frederic Alauzet, and Adrien Loseille. 2017. On a Robust Boundary Layer Mesh Generation Process. In *55th AIAA Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics, Grapevine, Texas. https://doi.org/10.2514/6.2017-0585

Naoki Marumo, Takayuki Okuno, and Akiko Takeda. 2020. Majorization-Minimization-Based Levenberg–Marquardt Method for Constrained Nonlinear Least Squares. https://doi.org/10.48550/arXiv.2004.08259 arXiv:2004.08259 [math]

Shigeyuki Morita. 2001. *Geometry of differential forms*. Number 201. American Mathematical Soc.

Luca Morreale, Noam Aigerman, Vladimir Kim, and Niloy J. Mitra. 2021. Neural Surface Maps. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Nashville, TN, USA, 4637–4646. https://doi.org/10.1109/CVPR46437.2021.00461

Patrick Mullen, Yiying Tong, Pierre Alliez, and Mathieu Desbrun. 2008. Spectral Conformal Parameterization. *Computer Graphics Forum* 27, 5 (2008), 1487–1494. https://doi.org/10.1111/j.1467-8659.2008.01289.x

Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. Robust Field-Aligned Global Parametrization. *ACM Transactions on Graphics* 33, 4 (July 2014), 1–14. https://doi.org/10.1145/2601097.2601154

Ashish Myles and Denis Zorin. 2012. Global Parametrization by Incremental Flattening. *ACM Transactions on Graphics* 31, 4 (Aug. 2012), 1–11. https://doi.org/10.1145/2185520.2185605

Ashish Myles and Denis Zorin. 2013. Controlled-Distortion Constrained Global Parametrization. *ACM Transactions on Graphics* 32, 4 (July 2013), 1–14. https://doi.org/10.1145/2461912.2461970

Rajkishore Nayak and Rajiv Padhye. 2017. *Automation in garment manufacturing.* Woodhead publishing.

M. Nieser, U. Reitebuch, and K. Polthier. 2011. CubeCover– Parameterization of 3D Volumes. *Computer Graphics Forum* 30, 5 (2011), 1397–1406. https://doi.org/10.1111/j.1467-8659.2011.02014.x

Barrett O'Neill. 2006. *Elementary Differential Geometry, Revised 2nd Edition.* Elsevier.

Jonathan Palacios and Eugene Zhang. 2007. Rotational Symmetry Field Design on Surfaces. *ACM Transactions on Graphics* 26, 3 (July 2007), 55–es. https://doi.org/10.1145/1276377.1276446

Nico Pietroni, Marcel Campen, Alla Sheffer, Gianmarco Cherchi, David Bommes, Xifeng Gao, Riccardo Scateni, Franck Ledoux, Jean Remacle, and Marco Livesu. 2023. Hex-Mesh Generation and Processing: A Survey. *ACM Transactions on Graphics* 42, 2 (April 2023), 1–44. https://doi.org/10.1145/3554920

Nico Pietroni, Marco Tarini, and Paolo Cignoni. 2010. Almost Isometric Mesh Parameterization through Abstract Domains. *IEEE Transactions on Visualization and Computer Graphics* 16, 4 (July 2010), 621–635. https://doi.org/10.1109/TVCG.2009.96

Ulrich Pinkall and Konrad Polthier. 1996. Computing Discrete Minimal Surfaces and Their Conjugates. *Experimental Mathematics* 2 (Nov. 1996). https://doi.org/10.1080/10586458.1993.10504266

Kacper Pluta, Michal Edelstein, Amir Vaxman, and Mirela Ben-Chen. 2021. PH-CPF: Planar Hexagonal Meshing Using Coordinate Power Fields. *ACM Transactions on Graphics* 40, 4 (July 2021), 156:1–156:19. https://doi.org/10.1145/3450626.3459770

Roi Poranne, Marco Tarini, Sandro Huber, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Autocuts: Simultaneous Distortion and Cut Optimization for UV Mapping. *ACM Transactions on Graphics* 36, 6 (Dec. 2017), 1–11. https://doi.org/10.1145/3130800.3130845

Lenka Ptáčková and Luiz Velho. 2021. A Simple and Complete Discrete Exterior Calculus on General Polygonal Meshes. *Computer Aided Geometric Design* 88 (June 2021), 102002. https://doi.org/10.1016/j.cagd.2021.102002

Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable Locally Injective Mappings. *ACM Transactions on Graphics* 36, 2 (April 2017), 1–16. https://doi.org/10.1145/2983621

Ernst Rank, Manfred Schweingruber, and Markus Sommer. 1993. Adaptive Mesh Generation and Transformation of Triangular to Quadrilateral Meshes. *Communications in Numerical Methods in Engineering* 9, 2 (1993), 121–129. https://doi.org/10.1002/cnm.1640090205

Nicolas Ray, Wan Chiu Li, Bruno Le Vy, and Pierre Alliez. 2006. Periodic Global Parameterization. *ACM Transactions on Graphics* 25, 4 (2006), 26.

Nicolas Ray, Vincent Nivoliers, Sylvain Lefebvre, and Bruno Lévy. 2010. Invisible Seams. *Computer Graphics Forum* 29, 4 (2010), 1489–1496. https://doi.org/10.1111/j.1467-8659.2010.01746.x

Nicolas Ray, Dmitry Sokolov, and Bruno Lévy. 2016. Practical 3D Frame Field Generation. *ACM Transactions on Graphics* 35, 6 (Nov. 2016), 1–9. https://doi.org/10.1145/2980179.2982408

Nicolas Ray, Bruno Vallet, Wan Chiu Li, and Bruno Lévy. 2008. N-Symmetry Direction Field Design. *ACM Transactions on Graphics* 27, 2 (May 2008), 10:1–10:13. https://doi.org/10.1145/1356682.1356683

J-F Remacle, C Geuzaine, G Compere, and E Marchandise. 2009. High Quality Surface Remeshing Using Harmonic Maps. (2009).

J.-F. Remacle, F. Henrotte, T. Carrier-Baudouin, E. Béchet, E. Marchandise, C. Geuzaine, and T. Mouton. 2013. A Frontal Delaunay Quad Mesh Generator Using the L ∞ Norm. *Internat. J. Numer. Methods Engrg.* 94, 5 (2013), 494–512. https://doi.org/10.1002/nme.4458

Ahmad A. Rushdi, Scott A. Mitchell, Ahmed H. Mahmoud, Chandrajit C. Bajaj, and Mohamed S. Ebeida. 2017. All-Quad Meshing without Cleanup. *Computer-Aided Design* 85 (April 2017), 83–98. https://doi.org/10.1016/j.cad.2016.07.009

Andrew O. Sageman-Furnas, Albert Chern, Mirela Ben-Chen, and Amir Vaxman. 2019. Chebyshev Nets from Commuting PolyVector Fields. *ACM Transactions on Graphics* 38, 6 (Dec. 2019), 1–16. https://doi.org/10.1145/3355089.3356564

Rohan Sawhney and Keenan Crane. 2018. Boundary First Flattening. *ACM Transactions on Graphics* 37, 1 (Jan. 2018), 1–14. https://doi.org/10.1145/3132705

Patrick Schmidt, Janis Born, Marcel Campen, and Leif Kobbelt. 2019. Distortion-Minimizing Injective Maps between Surfaces. *ACM Transactions on Graphics* 38, 6 (Dec. 2019), 1–15. https://doi.org/10.1145/3355089.3356519

Patrick Schmidt, Marcel Campen, Janis Born, and Leif Kobbelt. 2020. Inter-Surface Maps via Constant-Curvature Metrics. *ACM Transactions on Graphics* 39, 4 (Aug. 2020). https://doi.org/10.1145/3386569.3392399

Teseo Schneider, Yixin Hu, Xifeng Gao, Jérémie Dumas, Denis Zorin, and Daniele Panozzo. 2022. A Large-Scale Comparison of Tetrahedral and Hexahedral Elements for Solving Elliptic PDEs with the Finite Element Method. *ACM Transactions on Graphics* 41, 3 (June 2022), 1–14. https://doi.org/10.1145/3508372

Christian Schüller, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. 2013. Locally Injective Mappings. *Computer Graphics Forum* 32, 5 (2013), 125–135. https://doi.org/10.1111/cgf.12179

Nicholas Sharp and Keenan Crane. 2018. Variational Surface Cutting. *ACM Transactions on Graphics* 37, 4 (Aug. 2018), 1–13. https://doi.org/10.1145/3197517.3201356

RW Sharpe. 1997. Cartans Generalization of Kleins Erlangen Program. *Differential geometry, Graduate Texts in Mathematics* 166 (1997).

A. Sheffer and E. de Sturler. 2001. Parameterization of Faceted Surfaces for Meshing Using Angle-Based Flattening. *Engineering With Computers* 17, 3 (Oct. 2001), 326–337. https://doi.org/10.1007/PL00013391

A. Sheffer and J.C. Hart. 2002. Seamster: Inconspicuous Low-Distortion Texture Seam Layout. In *IEEE Visualization, 2002. VIS 2002.* IEEE, Boston, MA, USA, 291–298. https://doi.org/10.1109/VISUAL.2002.1183787

Alla Sheffer, Bruno Lévy, Maxim Mogilnitsky, and Alexander Bogomyakov. 2005. ABF++: Fast and Robust Angle Based Flattening. *ACM Transactions on Graphics* 24, 2 (April 2005), 311–330. https://doi.org/10.1145/1061347.1061354

Hanxiao Shen, Leyi Zhu, Ryan Capouellez, Daniele Panozzo, Marcel Campen, and Denis Zorin. 2022. Which Cross Fields Can Be Quadrangulated? Global Parameterization from Prescribed Holonomy Signatures. *ACM Transactions on Graphics* 41, 4 (July 2022), 59:1–59:12. https://doi.org/10.1145/3528223.3530187

Anna Shtengel, Roi Poranne, Olga Sorkine-Hornung, Shahar Z. Kovalsky, and Yaron Lipman. 2017. Geometric Optimization via Composite Majorization. *ACM Transactions on Graphics* 36, 4 (Aug. 2017), 1–11. https://doi.org/10.1145/3072959.3073618

Hang Si. 2015. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Trans. Math. Software* 41, 2 (Feb. 2015), 1–36. https://doi.org/10.1145/2629697

Yousuf Soliman, Albert Chern, Olga Diamanti, Felix Knöppel, Ulrich Pinkall, and Peter Schröder. 2021. Constrained Willmore Surfaces. *ACM Transactions on Graphics* 40, 4 (July 2021), 112:1–112:17. https://doi.org/10.1145/3450626.3459759

Yousuf Soliman, Dejan Slepčev, and Keenan Crane. 2018. Optimal Cone Singularities for Conformal Flattening. *ACM Transactions on Graphics* 37, 4 (Aug. 2018), 1–17. https://doi.org/10.1145/3197517.3201367

Justin Solomon, Keegan Crane, and Etienne Vouga. 2014. Laplace-Beltrami: The Swiss army knife of geometry processing. In *Symposium on Geometry Processing Graduate School (Cardiff, UK, 2014)*, Vol. 2.

Olga Sorkine and Marc Alexa. 2007. As-Rigid-As-Possible Surface Modeling. (2007), 8.

Boris Springborn, Peter Schröder, and Ulrich Pinkall. 2008. Conformal Equivalence of Triangle Meshes. *ACM Transactions on Graphics* 27, 3 (Aug. 2008), 1–11. https://doi.org/10.1145/1360612.1360676

B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. 2020. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation* 12, 4 (2020), 637–672. https://doi.org/10.1007/s12532-020-00179-2

Marco Tarini, Kai Hormann, Paolo Cignoni, and Claudio Montani. 2004. PolyCube-Maps. *ACM Transactions on Graphics* 23, 3 (Aug. 2004), 853–860. https://doi.org/10.1145/1015706.1015810

W. T. Tutte. 1963. How to Draw a Graph. *Proceedings of the London Mathematical Society* s3-13, 1 (1963), 743–767. https://doi.org/10.1112/plms/s3-13.1.743

Bruno Vallet and Bruno Lévy. 2009. What You Seam Is What You Get. (June 2009).

Amir Vaxman, Marcel Campen, Olga Diamanti, Daniele Panozzo, David Bommes, Klaus Hildebrandt, and Mirela Ben-Chen. 2016. Directional Field Synthesis, Design, and Processing. *Computer Graphics Forum* 35, 2 (2016), 545–572. https://doi.org/10.1111/cgf.12864

Ryan Viertel and Braxton Osting. 2018. An Approach to Quad Meshing Based on Harmonic Cross-Valued Maps and the Ginzburg-Landau Theory. arXiv:1708.02316 [cs, math]

Ke Wang. 2006. Edge Subdivision Schemes and the Construction of Smooth Vector Fields. (July 2006).

Y. Wang, B. Liu, and Y. Tong. 2012. Linear Surface Reconstruction from Discrete Fundamental Forms on Triangle Meshes. *Computer Graphics Forum* 31, 8 (2012), 2277–2287. https://doi.org/10.1111/j.1467-8659.2012.03153.x

Zhao Wang, Zhong-xuan Luo, Jie-lin Zhang, and Emil Saucan. 2016. ARAP++: An Extension of the Local/Global Approach to Mesh Parameterization. *Frontiers of Information Technology & Electronic Engineering* 17, 6 (June 2016), 501–515. https://doi.org/10.1631/FITEE.1500184

Colin Weill–Duflos and David Coeurjolly. 2023. Joint Optimization of Distortion and Cut Location for Mesh Parameterization Using an Ambrosio-Tortorelli Functional. (2023).

Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. 2019. Deep Geometric Prior for Surface Reconstruction. arXiv:1811.10943 [cs]

Yin Xu, Renjie Chen, Craig Gotsman, and Ligang Liu. 2011. Embedding a Triangular Graph within a given Boundary. *Computer Aided Geometric Design* 28, 6 (Aug. 2011), 349–356. https://doi.org/10.1016/j.cagd.2011.07.001

Dong-Ming Yan, Bruno Lévy, Yang Liu, Feng Sun, and Wenping Wang. 2009. Isotropic Remeshing with Fast and Exact Computation of Restricted Voronoi Diagram. *Computer Graphics Forum* 28, 5 (2009), 1445–1454. https://doi.org/10.1111/j.1467-8659.2009.01521.x

Rhaleb Zayer, Bruno Lévy, and Hans-Peter Seidel. 2007. Linear Angle Based Parameterization. (2007), 8.

Eugene Zhang, Konstantin Mischaikow, and Greg Turk. 2005. Feature-Based Surface Parameterization and Texture Mapping. *ACM Transactions on Graphics* 24, 1 (Jan. 2005), 1–27. https://doi.org/10.1145/1037957.1037958

Jiayi Eris Zhang, Alec Jacobson, and Marc Alexa. 2021. Fast Updates for Least-Squares Rotational Alignment. *Computer Graphics Forum* 40, 2 (2021), 13–22. https://doi.org/10.1111/cgf.142611

Muyang Zhang, Jin Huang, Xinguo Liu, and Hujun Bao. 2010. A Wave-Based Anisotropic Quadrangulation Method. *ACM Transactions on Graphics* 29, 4 (July 2010), 1–8. https://doi.org/10.1145/1778765.1778855

Paul Zhang, Josh Vekhter, Edward Chien, David Bommes, Etienne Vouga, and Justin Solomon. 2020. Octahedral Frames for Feature-Aligned Cross Fields. *ACM Transactions on Graphics* 39, 3 (June 2020), 1–13. https://doi.org/10.1145/3374209