ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

Bert Buchholz

Rapport de Thèse
Futur et Rupture

–

Mi-Parcours

DIRECTEURS
Tamy Boubekeur
Isabelle Bloch

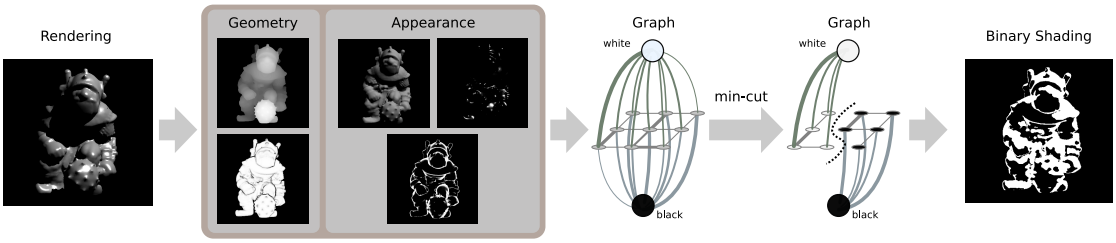

Figure 1: Les étapes de "Binary Shading".

1 Binary Shading

1.1 Introduction

Cette partie de ma thèse est la suite de mes travaux de stage Master sur le sujet "Hybrid Binary Shading". La génération des images avec forte réduction du nombre de couleurs est motivée par des usages artistiques, pour faire de la stylisation, pour respecter les contraintes d'écran ou pour une meilleure représentation visuelle. Dans notre cas c'est une réduction jusqu'à deux couleurs qui donne des image binaires.

Il y a quelques techniques qui génèrent des images binaires en dessinant des lignes ([JDA07], [DFRS03]). Ces techniques partent normalement d'une scène 3D et utilisent la géométrie de l'objet et la position de la caméra pour déterminer quelles lignes à dessiner. Les autres techniques partent souvent d'une image 2D (par exemples des photographies) et les couleurs sont réduites en utilisant differentes algorithmes. [XKM07] génère des images binaires dans le but de pouvoir les couper (silhouette), donc ils ont besoin de grandes regions colorées de façon cohérente qui sont toutes connectées. Pour des raisons artistiques, [XK08] créent des images binaires en utilisant que du seuillage sophistiqué. [MG08] proposent une technique un peu plus coûteuse qui utilise entre autres le Graph Cut ([BJ01]) également utilisé dans notre méthode pour générer l'image binaire.

Partant d'une scène 3D, le but de l'approche de "binary shading" est de créer des images composées de deux couleurs (habituellement blanc et noir) constituées de larges regions colorées de façon cohérente. Cette technique est une méthode de Rendu Non Photoréaliste (NPR) qui génère des résultats qui ne sont pas physiquement possibles.

1.2 Algorithme

L'algorithme se décompose en deux parties principales: La première, consiste à obtenir des informations sur la scène. La deuxième, est d'utiliser ces informations pour générer un graphe qui est segmenté en deux ensembles représentants les deux couleurs de l'image.

Dans la première partie, les informations sont genérées par raytracing de la scène à partir d'un point de vue et une resolution données par l'utilisateur. Le résultat est une image 2D qui contient un vecteur de deux genres principaux d'informations sur la scène. Pour chaque pixel contenant une intersection avec la géométrie de la scène, on conserve des informations

sur l'apparence et la géométrie de la surface en ce point. Un pixel correspond bien sûr à une position dans l'espace de l'image 2D mais il correspond également à une position 3D sur la surface de l'objet.

Il y a plein de differentes propriétés de l'apparence et de la géométrie que l'on peut utiliser. Dans ce travail, on utilise pour l'apparence la lumière diffuse et spéculaire (qui dépendent d'au moins une source de lumière dans la scène) et la silhouette de l'objet (derivée de l'angle entre le rayon de vue et la normale de la surface à chaque pixel).

L'information géométrique concerne la géométrie de la surface de l'objet. On utilise la courbure locale de la surface et la position de la surface à chaque pixel. La courbure locale est mesurée par la difference entre les normales dans une petite région autour d'un point de la surface. Cela permet une estimation des discontinuités sur la surface et donc de trouver les limitations naturelles des regions sur la surface, donc des caractéristiques (features) comme bumps, ridges et valleys. L'information sur la position est utilisée pour trouver les discontinuités de la distances entre deux pixels qui peuvent être voisins dans l'espace de l'image mais qui sont très eloignés dans l'espace 3D.

Ces informations sont rassemblées dans une grille d'une image 2D dans laquelle chaque pixel contient l'information sur l'apparence et la géométrie du point de la surface qui correspond à ce pixel. Cette image 2D est convertie en graphe dont chaque sommet correspond à un pixel. Même si la connectivité entre les sommets n'est pas fixe, le 8-voisinage est habituellement utilisé, donc chaque sommet est connecté à ses voisins directs horizontaux, verticaux et diagonaux. Chaque sommet est aussi connecté à deux sommets spéciaux (appelés terminaux, terminals en anglais), source et sink (en anglais). Le but est de couper le graphe en deux ensembles disparates où chaque sommet n'est connecté qu'à un seul de ces deux sommets spéciaux (soit directement, soit indirectement).

La coupe même est trouvée en resolvant le problème min-cut/max-flow pour lequel on cherche à trouver la coupe qui minimise la somme des coûts des arêtes coupées. Le coût de coupe d'une arête est sa capacité. Donc, en général, plus une capacité est élevée, plus la probabilité que l'arête ne soit pas coupée est élevée. Mais comme la solution est globale, la coupe finale dépend de toutes les arêtes. On utilise la méthode Graph Cut ([BJ01], [BK04]) qui est très répendue depuis quelques années dans le domaine de la segmentation d'image.

Les capacités entre les sommets normaux (capacité de voisins) sont dérivées à partir de l'information géométrique. Comme le but, mentionné ci-dessus, est de trouver des regions larges de blanc et noir, on essaie de remplir les regions de l'objet qui sont géométriquement cohérentes avec la même couleur. Pour y arriver, on utilise le fait que des differences basses de la courbure et des petites discontinuités de la profondeur sont présentes aux grandes capacités des arêtes. Donc, des regions par exemple plates (et donc cohérentes) ont une forte probabilité de rester connectées.

Les capacités entre chaque sommet et les deux terminaux (capacités terminales) sont déterminées à partir de l'information d'apparence. Une valeur élevée de l'apparence va résulter dans une capacité élevée vers la source et basse vers le sink et inversement.

Les capacités terminales peuvent être considerées comme une tendance à une de les deux terminales (et donc à une couleur) pour chaque pixel individuel. Les capacités de voisins servent à la préservation de la cohérence des regions.

Après l'exécution d'une coupe respectant ces contraintes (trouver un coupe qui sépare les deux sommets terminaux), les sommets du graphe sont reconvertis en une image 2D où à chaque sommet connecté au sommet terminal sink est attribué une couleur (normalement blanc) et à chaque sommet connecté au sommet terminal source est attribué l'autre couleur (normalement noir), donnant ainsi l'image finale de deux couleurs.

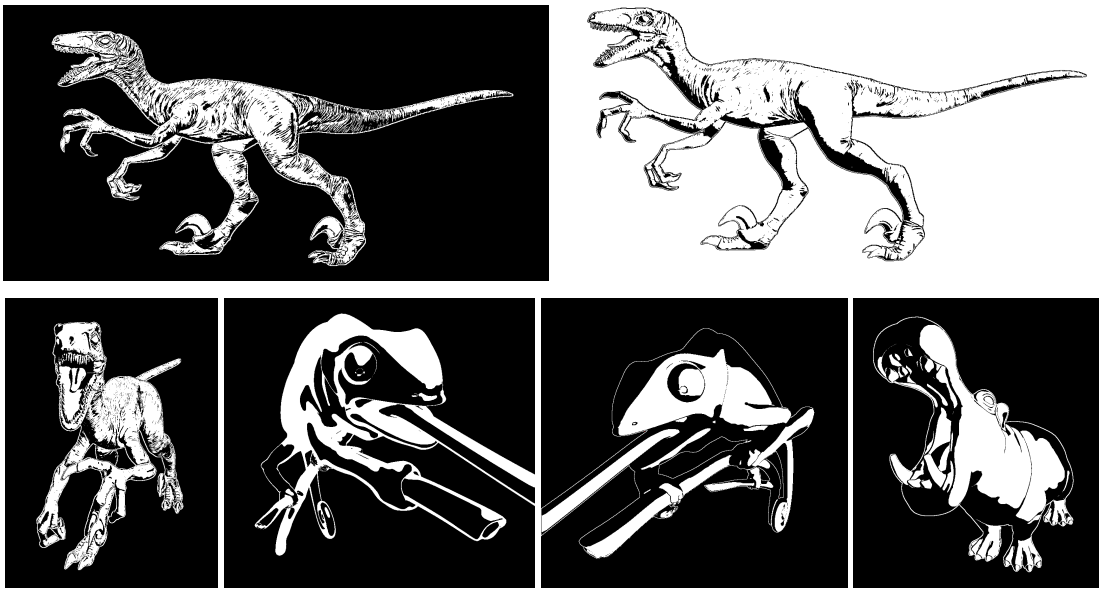

Figure 2: Les résultats de la technique "binary shading".

1.3 Résultats

L'utilisateur peut influencer les résultats finaux en changeant des poids associés à chaque type d'information de l'apparance et géométrie. En agissant sur ces poids, l'utilisateur peut influencer directement et indirectement les capacités dans le graphe qui déterminent le résultat final. Ce travail est paru dans [BBDA10].

2 Temporally Consistent Line Drawings

2.1 Introduction

La paramétrisation de lignes de manière temporellement cohérente est interessant pour de nombreux domaines mais particulièrement pour les dessins animés. Dans les dessins animés

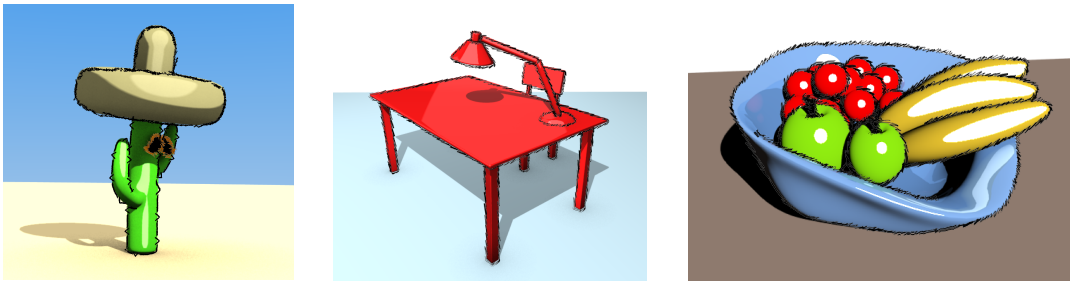

Figure 3: Des trames extraites de trois animations montrants les contours stylisés avec la technique "Temporally Consistent Line Drawings".

générés par ordinateur, les lignes sont normalement les contours des objets de la scène. Souvent, ces lignes ne sont pas seulement dessinées par un trait de la même couleur et la même épaisseur, mais par des traits variants (en utilisant des textures) pour ressembler à des dessins d'un artiste humain. Pour cela, une paramétrisation des lignes est nécessaire. Dans ce cas, il faut trouver un moyen de forcer une certaine cohérence temporelle, car si chaque ligne est paramétrée indépendamment pour chaque image d'une animation, les textures sur les lignes semblent glisser sur la surface de l'objet. Avec une paramétrisation temporellement cohérente, la texture semble être attachée à la surface.

De nombreux travaux existent sur le sujet de l'extraction de lignes à partir d'un objet 3D ([JDA07], [DFRS03]), mais normalement le but n'est pas la paramétrisation. Quelques travaux s'intéressent également à la cohérence temporelle des lignes mais pas à leur paramétrisation ([DFR04]). La méthode, présentée ici, suit les travaux de Kalnins ([KDMF03]) qui a le même objectif mais propose une technique temps réel. Celle-ci ne peut donc pas prendre en compte la partie de l'animation future et ainsi ne peut pas trouver une solution globalement optimale mais juste une solution d'une trame à l'autre.

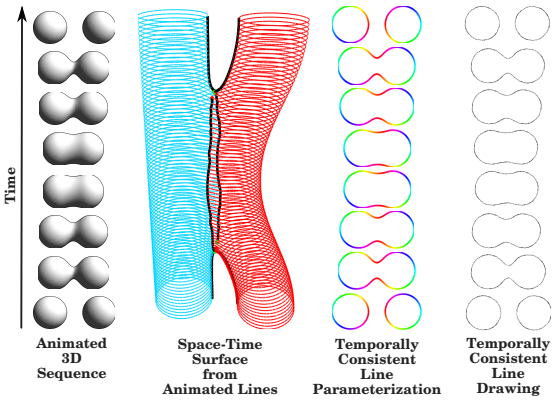

Figure 4: Les étapes differentes de la paramétrisation cohérente des lignes.

2.2 Algorithme

Notre algorithme est composé de deux parties. Etant donné est un ensemble des lignes que l'on a obtenu avant. Par exemple, dans le cas d'une animation 3D, les lignes peuvent être les contours de l'objet 3D. Les lignes qui sont paramétrées à la fin sont les projections des ces con-

tours 3D. Chaque trame de l'animation contient quelques lignes qui ont des correspondances dans les trames précédentes et suivantes. Dans une première phase de l'algorithme, il faut trouver ces correspondances temporelles. Une fois les correspondances établies, les groupes des lignes correspondantes sont paramétrées indépendantement des autres lignes.

Pour établir la cohérence temporelle de la paramétrisation, on utilise plusieurs mesures d'erreur comme contraintes et on utilise la méthode des moindres carrés pour minimiser l'erreur. Pour chaque sommet d'une ligne, on essaie de trouver une valeur de paramétrisation qui satisfait le mieux les contraintes suivantes:

- Conserver une valeur de zéro au début de la ligne et une valeur de un à la fin de la ligne
- Garder une distance uniforme entre les valeurs de paramétrisation des sommets qui correspond à leur distance après la projection
- La valeur de paramétrisation d'une partie de la ligne doit suivre le mouvement de la ligne. Il s'ensuit la cohérence temporelle.

On peut noter que les deux dernières contraintes se contredisent. N'avoir que la contrainte uniforme va aboutir au problème du glissement mentionné ci-dessus. L'autre extrème, n'avoir que la contrainte temporelle, va aboutir à l'étirement des valeurs et la texture peut être transformée d'une manière non voulue. En conséquence il faut trouver un équilibre entre les deux, mais comme cet équilibre est plutôt de nature esthétique, le mélange final est cédé à l'utilisateur.

3 Parameterized Hexahedral Meshing

Il existe plusieurs représentations volumiques d'un maillage 3D, notamment la méthode utilisant une grille regulière 3D (hexaèdres) alignée avec les axes à l'intérieur et une subdivision plus fine près de la surface approximée. Une autre méthode est la tétraédrisation, qui rempli le maillage avec des tétraèdres sans avoir à approximer la surface ([TWAD09]). Le désavantage de la tétraédrisation par rapport à la hexaédrisation, est qu'il n'y a pas d'ordre à l'intérieur du maillage. Notamment dans le but d'appliquer des traitements interactifs et de paramétrer l'intérieur, cela peut être problématique.

Donc, l'idée est de trouver une représentation volumique qui soit dérivée de la surface et qui continue vers l'intérieur d'une manière facilement compréhensible pour faciliter l'utilisation interactive. Dans un premier temps, les sommets de la surface sont contractés vers l'intérieur jusqu'à on arrive à un squelette de zéro volume consistant en un ensembles de lignes ou de pellicules. La façon exacte pour y arriver n'est pas cruciale, la seule contrainte est que les trajectoires des sommets ne se croisent pas. Ensuite, ces trajectoires sont subdivisées en un nombre de segments donné par l'utilisateur. Ces segments déterminent des couches de volume dans l'intérieur de l'objet. Chaque élément de volume est le volume compris entre deux couches voisines et les trajectoires voisines (voir fig. 5).

La paramétrisation de la surface, en liaison avec la paramétrisation de trajectoires, donne une paramétrisation de l'intérieur. Comme la structure des couches est dérivée directement

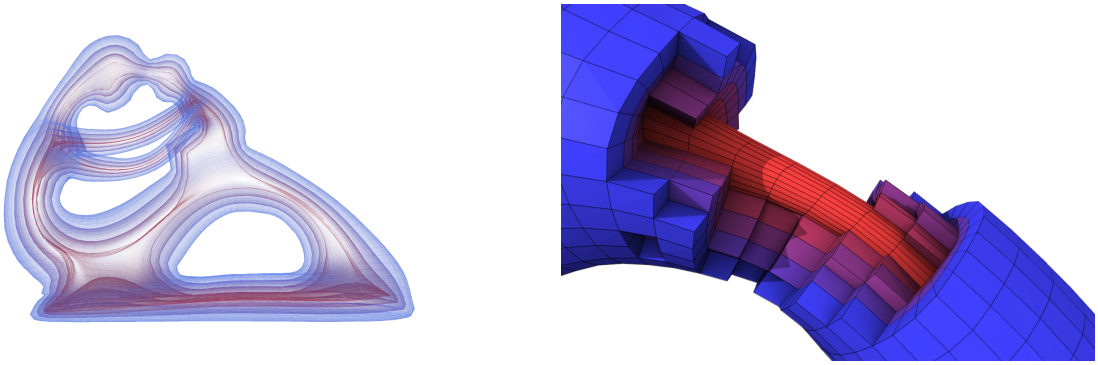

Figure 5: Les couches dans une représentation transparente (gauche) et une coupe dans l'intérieur d'une partie du maillage (droit).

de la surface et d'un squelette, on attend à ce que cela va faciliter les opérations de traitement volumique. Les travaux décrits dans cette section sont encore en cours.

4 Subsurface Scattering

Figure 6: Exemple d'un résultat qui n'est pas physiquement correct: la lumière qui traverse le nez est trop forte à la base et trop faible à la pointe (résultat de [DS03]).

Ce travail est un projet avec mon collègue Matthias Holländer et n'est pas encore fini. La transluminescence (subsurface scattering, SSS) décrit le phénomène optique quand la lumière traverse des objets translucides (mais pas transparents) dans lesquels la lumière interagit avec le milieu matériel et est dispersée à l'intérieur de celui-ci. La dispersion signifie que la lumière ne traverse pas forcément l'objet complètement ou sur une ligne droite. Chaque rayon de la lumière qui pénètre dans l'objet peut être dans certains cas complètement absorbée ou être diffusée avant ressortir.

A l'heure actuelle, il existe deux types de rendu de cet effet. L'un n'est pas assez physiquement correct et donc produit des résultats faux dans de nombreuses situtations (voir fig. 6), mais cette technique est assez peu coûteuse en temps de calcul et donc son usage se justifie lorsque l'on souhaite obtenir cet effet sans y consacrer trop de temps de calcul. L'autre technique est beaucoup plus exacte mais en même temps beaucoup plus coûteuse ([WWH$^+$10]). Les précalculs sont souvent nécessaires et dépendent normalement de la complexité de la scène. Contrairement à l'autre technique, celle-ci peut être physiquement cor-

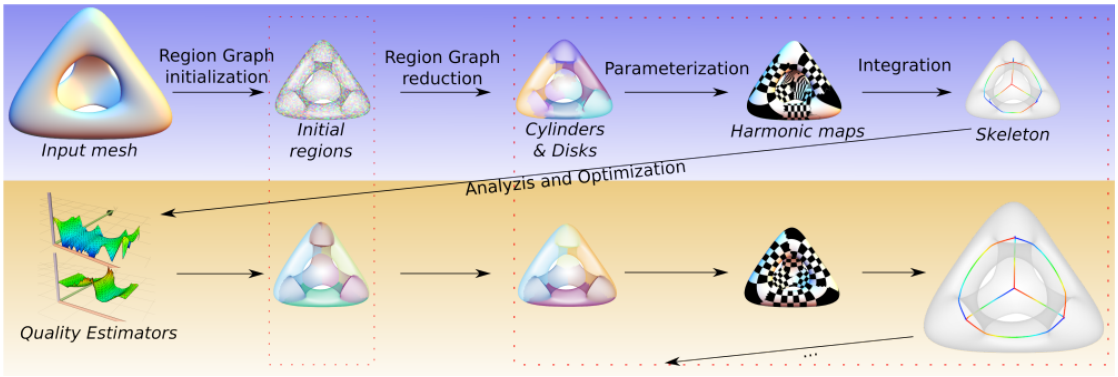

Figure 7: Les étapes de "Euler Skeleton".

rect mais coûte trop cher pour être utilisé dans des applications qui ne peuvent pas dédier beaucoup de temps de calcul pour cet effet.

Notre idée s'approche à la méthode proposée dans [DS03]. Leur méthode est de premier type et il y a quelques cas dans lesquels cette méthode donne des résultats incorrects, visibles sur leurs images. Notre but est de trouver une méthode qui soit aussi rapide mais qui offre une meilleure solution dans les cas problématiques.

Pour calculer SSS en un certain point p vu par la caméra, il faut intégrer la lumière qui entre dans l'objet en n'importe quel point, simuler la diffusions des rayon de lumière dans le milieu matériel et calculer la quantité de lumière diffusée qui sort l'objet au point p dans la direction de la caméra. Pour simplifier cette calcul, seule la lumière arrivant directement à la surface par le rendu de *light map* de la lumière (directionelle) est prise en compte. Pour chaque pixel on trouve la première couche de volume qui est representée par l'espace entre la première est la deuxième profondeur. C'est-à-dire, le première point où le rayon de la caméra entre dans l'objet et le point où ce rayon sort de l'objet.

Notre première idée était de calculer la quantité de la lumière qui est diffusée à partir d'une certaine région autour du point de sortie (comme il est vu par la lumière) vers l'intérieur de l'objet le long du rayon qui correspond au pixel rendu. Nous avons mené des expériences avec differentes quantités de points distribués entre les points d'entrée et de sortie mais les résultats ne sont pas satisfaisants pour l'instant.

5 Euler Skeleton

Pour ce projet j'ai été impliqué surtout pendant la première étape du développement du algorithme. La part la plus importante a été réalisé par mon collègue Jean-Marc Thiery. Le résultat de ces travaux été accepté en tant que poster à la conférence Symposium on Geometry Processing (SGP) 2010.

Le but de ces travaux est de générer une paramétrisation de la surface d'un objet en segmentant la surface en cylindres et disques et utiliser ensuite la paramétrisation standard de ces

formes géométriques et de l'appliquer à la surface. Ainsi, on obtient la paramétrisation de tous les segments de la surface en termes de cylindres et de disques. J'ai principalement contribuer pour trouver un moyen de faire la segmentation (voir fig. 7, jusqu'à "Parameterization").

La méthode commence par segmenter toute la surface en petits morceaux (surface patches) si nécessaire jusqu'au niveau des triangles. Ensuite, les morceaux voisins sont rejoints dans le but de former des cylindres et disques en utilisant des règles et contraintes guidant ces unions pour y arriver. Pour chaque union possible entre deux morceaux, on trouve une valeur de priorité en comparant des propriétés géométriques de ces deux morceaux avec le morceau resultant de l'union. Plus un morceau joint ressemble à un cylindre, plus il est important et plus est l'union probable. Après quelques itérations les premièrs cylindres sont formés et sont ensuite unis en priorité avec autres cylindres voisins.

Une fois que toutes les regions sont soit des cylindres soit des disques (c'est-à-dire des disques qui ne peuvent pas être joint avec un cylindre sans transformer le cylindre en disque), la paramétrisation des cylindres est ensuite faite. On considère les cylindres comme os et on en dérive un squelette avec un os par cylindres et des articulations où deux os ou plus se touchent. A partir de ce squelette, on mesure la qualité de la rélation entre le squelette et la surface et on utilise le squelette pour refaire la segmentation. A ce stade, les étapes dernières peuvent être répétées pour améliorer la segmentation et en même temps la paramétrisation ainsi que le squelette.

References

[BBDA10] Bert Buchholz, Tamy Boubekeur, Doug DeCarlo, and Marc Alexa. Binary shading using geometry and appearance. 29(6):1981–1992, 2010.

[BJ01] Y. Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *ICCV*, pages I: 105–112, 2001.

[BK04] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE trans. on PAMI*, 26(9):1124–1137, Sept. 2004.

[DFR04] Doug DeCarlo, Adam Finkelstein, and Szymon Rusinkiewicz. Interactive rendering of suggestive contours with temporal coherence. In *NPAR 2004*, pages 15–24, June 2004.

[DFRS03] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape. *ACM Trans. Graph.*, 22(3):848–855, 2003.

[DS03] Carsten Dachsbacher and Marc Stamminger. Translucent shadow maps. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 197–201, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[JDA07] Tilke Judd, Frédo Durand, and Edward H. Adelson. Apparent ridges for line drawing. *ACM Trans. Graph.*, 26(3):19, 2007.

[KDMF03] Robert D. Kalnins, Philip L. Davidson, Lee Markosian, and Adam Finkelstein. Coherent stylized silhouettes. *ACM Transactions on Graphics*, 22(3):856–861, July 2003.

[MG08] David Mould and Kevin Grant. Stylized black and white images from photographs. In *NPAR*, pages 49–58, 2008.

[TWAD09] Jane Tournois, Camille Wormser, Pierre Alliez, and Mathieu Desbrun. Interleaving delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Trans. Graph.*, 28(3):1–9, 2009.

[WWH+10] Yajun Wang, Jiaping Wang, Nicolas Holzschuch, Kartic Subr, Jun-Hai Yong, and Baining Guo. Real-time rendering of heterogeneous translucent objects with arbitrary shapes, 2010.

[XK08] Jie Xu and Craig S. Kaplan. Artistic thresholding. In *Proc. of NPAR)*, pages 39–47, 2008.

[XKM07] Jie Xu, Craig S. Kaplan, and Xiaofeng Mi. Computer-generated papercutting. In *Pacific Graphics*, pages 343–350, 2007.