Simplification et abstraction de dessins au trait

Pascal Barla, Joëlle Thollot, François X. Sillion

Simplification et abstraction de dessins au trait

P. Barla, J. Thollot, F. Sillion

ARTIS / GRAVIR-IMAG-INRIA

pascal.barla,joelle.thollot,francois.sillion@imag.fr

Résumé : *Dans cet article, nous proposons une méthode pour simplifier un ensemble de lignes vectorielles tout en conservant la structure du dessin de départ et en incorporant des choix de style de la part de l'utilisateur. Cette approche a pour avantage d'être assez modulaire pour s'adapter à de nombreuses applications : de l'édition interactive de tracés à la génération de niveaux de détail pour le rendu non-photoréaliste, en passant par la gestion de la densité de dessins. Nous présentons ainsi un cadre commun à l'ensemble de ces méthodes et illustrons son potentiel par le biais de deux applications : un outil de tracé progressif et une méthode de mise à l'échelle d'un dessin.*

Mots-clés : lignes, simplification, *oversketching*, rendu non-photoréaliste, niveaux de détail, *beautification*, dessin au trait

1 Introduction

Enoncé du problème

Le travail présenté ici concerne le traitement de dessins au trait, que nous représentons comme des images numériques composées de lignes vectorielles. De telles images peuvent provenir de différentes sources : d'un dessin réalisé « à la main », puis scanné et converti par un algorithme d'extraction de lignes ; d'un dessin réalisé numériquement par le biais d'un périphérique (souris, tablette graphique, etc) ; d'une détection de contours dans une image [ZT98] ; d'un rendu au trait d'une scène 3D [SS02] [GG01] [GDS04] ; etc.

Parmi les nombreuses opérations disponibles sur de tels ensembles de lignes, nous nous intéressons en particulier aux méthodes de simplification. Par simplification, nous entendons la création d'un ensemble de lignes qui contient moins de lignes que l'ensemble initial. On retrouve cette problématique présentée dans différents contextes dans la littérature, notamment :

- L'édition progressive de tracés (aussi appelé « oversketching ») où l'utilisateur précise le tracé d'une courbe par retouches successives. Dans de nombreux cas, cette approche est nettement plus naturelle que l'édition de points de contrôle, notamment dans les systèmes de modélisation à base de croquis. Un workshop Eurographics spécialisé a démarré en 2004 [sbm04] et offre un bon aperçu des différentes applications de la modélisation à base de dessins ;
- La gestion de la densité des lignes d'un dessin : que ce soit lors de la mise à l'échelle d'un dessin au trait ou en post-traitement d'un rendu non-photoréaliste à base de contours d'une scène 3D, il peut arriver que trop de lignes soient présentes dans un même dessin, nuisant à sa compréhension. On cherche donc à mesurer la densité de ces ensembles de lignes afin de n'en afficher que les plus significatives, en tenant compte éventuellement d'informations externes (comme les informations présentes dans le rendu d'une scène 3D [GDS04]) ;
- Les niveaux de détail pour le rendu au trait (contours et hachures) où l'on cherche à diminuer le nombre de lignes en fonction de la distance à l'objet représenté. Les méthodes actuelles [KMM+02][PHWF01][WPFH02] sont plus ou moins automatiques et posent les problèmes classiques de la création et de la transition entre niveaux de détails. On s'intéresse alors à caractériser les liens entre les lignes de ces différents niveaux de détail, afin d'assister l'utilisateur dans leur conception d'une part ; et d'autre part, afin de proposer différentes stratégies à employer lors de la transition entre deux niveaux.

Malgré leurs spécificités, ces problématiques ont une même finalité : la création d'un ensemble de lignes plus petit que l'ensemble initial, selon un but précis. Dans le cas de l'édition progressive, ce but est guidé par un souci d'ergonomie, ce qui se traduit généralement par un ensemble d'heuristiques et de règles implicites que l'utilisateur doit avoir à l'esprit. Pour la gestion de la densité, le but est de créer les lignes les plus significatives tout en respectant une densité maximum dans l'image. Quant aux niveaux de détail, ce sont soit les choix esthétiques de l'utilisateur qui guident la création de ces lignes lorsque lui revient la charge de dessiner chaque niveau, soit,

dans certains cas précis comme les groupes de hachures, des règles de conservation du ton de l'image résultat qui guident la simplification.

Nous décrivons ici un cadre commun à ces différentes problématiques. Notre approche nécessite un unique paramètre ϵ afin de contrôler le processus complet de simplification. Il peut représenter, au choix, la taille d'une zone d'influence, un seuil de densité ou une échelle caractéristique. Elle est applicable dans de nombreux contextes puisqu'aucune contrainte n'est imposée sur la forme ou la nature des lignes en entrée.

Travaux précédents

Nous passons ici en revue les différents travaux ayant nécessité une simplification de dessin.

Les outils de dessin progressif [IMKT97] sont de plus en plus utilisés, notamment dans le cadre de la modélisation à partir de croquis [sbm04] ou dans les outils de dessin vectoriel comme Adobe Illustrator©. Ils proposent d'assister l'utilisateur dans le tracé d'une ligne par une série d'outils dédiés. Ces méthodes sont par nature semi-automatiques et sont uniquement destinées à l'édition progressive : elles sont difficilement adaptables à d'autres applications car elles requièrent trop d'interventions de la part de l'utilisateur et se basent sur des règles spécifiques.

Des algorithmes de traitement de densité de lignes provenant de rendus 3D ont déjà été présentés. Deussen et al. [DS00] proposent une simplification dédiée à la végétation et aux arbres. Grace à une structure hiérarchique, certains groupes d'objets, comme les feuilles, sont remplacés par des primitives plus simples. De plus, un seuil sur le z-buffer leur permet de ne rendre que les arêtes indiquant une grande discontinuité de profondeur. Cette approche permet une simplification puissante mais repose sur la représentation hiérarchique de la végétation et sur le z-buffer. Wilson et. al [WM04] mesurent une densité en espace image afin de limiter le nombre de lignes à dessiner pour des objets complexes. Grabli et. al [GDS04] présentent eux aussi une mesure de différentes densités en espace image qu'ils utilisent pour sélectionner les lignes les plus significatives à conserver et supprimer les autres. Leurs mesures sont plus variées et permettent donc de plus nombreux effets que la méthode de Wilson et. al. Ils tirent notamment parti de nombreuses informations provenant de la scène 3D, leur permettant de donner une importance aux lignes. Ces approches reposent cependant fortement sur la nature sous-jacente des lignes et dépendent donc du contexte du dessin ; elles nécessitent le réglage de plusieurs paramètres par l'utilisateur et s'exécutent ensuite automatiquement.

Dans le cas des images d'illustration, Winkenbach et al. [WS94] ont introduit la notion d'indication où des textures complexes ne sont dessinées complètement qu'en certains endroits pour suggérer la complexité d'un motif (un mur de briques par exemple) sans surcharger le dessin. Cette étude montre bien que le traitement de la densité des lignes d'un dessin dépend d'informations spatiales mais peut être augmenté d'informations sémantiques sur l'importance de certaines lignes ou de certaines zones de l'image.

Enfin, dans le cadre du rendu non-photoréaliste, divers articles proposent des méthodes adaptées au traitement des niveaux de détail pour le rendu d'objets en mouvement. Praun et al. [PHWF01] présentent une méthode à base d'images pour créer des niveaux de détail de hachures : ils créent des séries de textures MipMap correspondant à différentes tonalités de dessin, qu'ils appellent des Tonal Art Maps (TAMs), et qu'ils utilisent pour afficher en temps réel des objets 3D dans un style hachuré. Les TAMs sont créées procéduralement, ce qui limite les possibilités de création du côté de l'utilisateur. Une approche où les TAMs seraient créées automatiquement à partir d'une version exemple dessinée par l'utilisateur serait un autre moyen plus intuitif de les obtenir. D'autres systèmes de création de niveaux de détail ont été présentés récemment ; un module a été notament intégré au système WYSIWYG NPR [KMM+02] qui permet à l'utilisateur de spécifier directement les niveaux de détail à la surface des objets pour différents points de vue. Pour un point de vue donné, les niveaux de détail les plus pertinents sont combinés par un simple *blending*, créant une transition assimilable à un fondu enchainé. D'autres approches de transition peuvent être envisagées, tirant parti des liens entre les lignes situées sur deux niveaux de détail successifs.

Le même genre de problématique se retrouve lorsque l'on veut maitriser la résolution de l'image en vue par exemple d'une impression. Toutes ces méthodes sont reliées aux techniques de *half-toning*. Des travaux ont été effectués dans le cas des hachures, que ce soit à partir d'images [SALS96] ou de modèles 3D [ZISS04]. Là encore certains auteurs ont ajouté une notion de priorité sur les lignes qui permet de construire les différents niveaux de ton en dessinant d'abord les lignes les plus importantes.

A notre connaissance, il n'existe pas de méthode visant à mettre en relation ces diverses approches.

Contributions

Notre principale contribution est la définition d'un cadre commun aux processus de **simplification** d'un ensemble de lignes vectorielles. Pour cela nous décomposons le processus de création d'une version simplifiée d'un dessin en deux étapes (cf Figure 1) :

– Une étape de simplification dans laquelle nous allons choisir comment grouper les lignes du dessin d'origine de telle sorte que chaque groupe puisse être représenté par une seule ligne dans le dessin final. Plus précisement cette étape repose sur la définition d'une **structure de ligne à ϵ près** nous donnant les regroupements possibles et d'un **algorithme de simplification** modulaire.

– Une étape d'abstraction dans laquelle nous choisissons pour chaque groupe de lignes, la ligne finale à dessiner parmi toutes celles possibles. Cette étape est un choix de style et peut s'adapter à diverses applications. Nous présentons ici un **outil de tracé progressif**, et une méthode de **mise à l'échelle d'un dessin**. De nombreux autres outils sont envisageables, comme nous l'avons déjà évoqué et continuerons à le faire dans les prochaines sections.

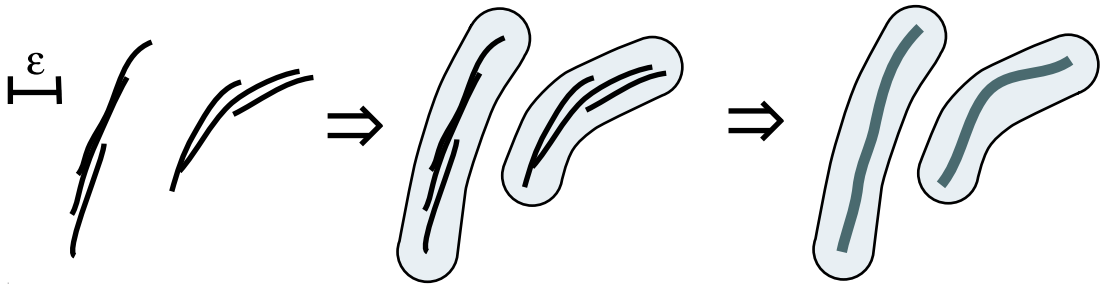

FIG. 1 – Les lignes en entrée (**à gauche**) sont d'abord regroupées pour former une partition de l'ensemble initial (**au milieu**). Chaque groupe de lignes est ensuite remplacé par une nouvelle ligne dans l'étape d'abstraction (**à droite**).

Nous commençons par décrire la méthodologie générale dans la section 2 ; puis dans les sections suivantes (3 et 4), nous présentons les deux étapes composant notre approche : la simplification et l'abstraction ; enfin nous détaillons l'implémentation et montrons les premiers résultats obtenus dans les sections 5 et 6.

2 Méthodologie

On part d'un dessin représenté par un ensemble de lignes initial que l'on note L. On veut créer un ensemble de lignes L', contenant moins de lignes que L et qui va représenter L à une précision donnée. Cette précision, que l'on note ϵ, correspond en pratique à une distance euclidienne en espace image. Toute information présente à une échelle inférieure à ϵ est dès lors considérée comme du détail et pourra être éventuellement simplifiée dans la suite.

On commence par regrouper les lignes de L en sous-ensembles distincts ; les lignes de L' seront ensuite générées à partir de ces groupes, une ligne par groupe. On crée donc une **partition** π de L où chaque partie est considérée comme une seule et même ligne à ϵ près. Etant donné que notre but principal est de conserver la structure du dessin avec un nombre réduit de lignes, on impose une contrainte sur chaque élément de π : chaque groupe de ligne ainsi formé doit respecter la **structure d'une ligne à ϵ près** (voir section 2.1). La première étape de notre algorithme, la **simplification**, consiste à trouver une partition « optimale » de L, π^*, qui va comporter le moins possible de groupe tout en conservant le niveau de simplification voulu donné par ϵ. Nous définissions pour cela une mesure d'erreur à la section 3.1.

On veut ensuite générer une ligne de L' à partir de chaque élément de π^*. Afin de respecter la structure de ligne à ϵ près imposée à chacun des groupes de lignes, on contraint les lignes générées à demeurer « à l'intérieur » de leur groupe. En pratique, on impose une **distance de Hausdorff inférieure à** ϵ entre la ligne générée et son groupe associé. Cette contrainte laisse cependant de nombreux degrés de liberté à l'utilisateur et diverses stratégies peuvent être employées pour la génération de L'. La second étape, la phase d'**abstraction**, va ainsi consister à choisir une stratégie. On discutera des besoins de chaque type d'application et leur impact sur le choix des stratégies à employer.

2.1 Structure de ligne à ϵ près

On veut conserver la structure du dessin initial dans le nouveau dessin simplifié. Pour un ensemble de lignes E du dessin initial, on veut donc que la nouvelle ligne qui le représente, l', conserve sa structure. On montre dans la Figure 2 un exemple d'association valide et un exemple d'association invalide. Afin d'aboutir à une association valide, on impose que E soit assimilable à une ligne à ϵ près.

Plus formellement, on dit qu'un ensemble de lignes E possède une structure de ligne à ϵ près ssi :

– E a exactement deux *extrémités à ϵ près* ;
– il existe un *chemin unique à ϵ près* entre ces deux extrémités.

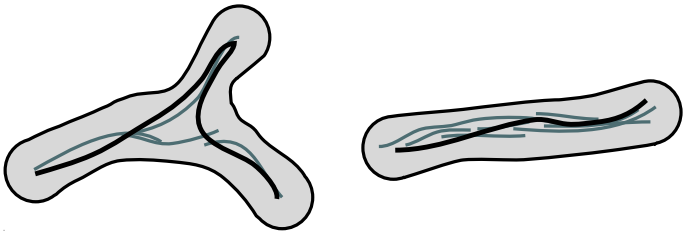

FIG. 2 – **A gauche**, la ligne créée est bien contenue dans la dilatation de l'ensemble des lignes par ϵ, mais elle ne respecte pas sa structure ; contrairement au dessin **de droite**, où la ligne créée conserve la structure sous-jacente.

On commence par définir une *extrémité à ϵ près* dans E.

Définition : un point p^* sur une ligne de E est considéré comme une extrémité à ϵ près de E ssi l'ensemble des points de E à une distance ϵ de p^* peut être contenu dans un disque de diamètre ϵ (ils sont tous « du même côté de p^* ») :

$$\exists B_\epsilon \text{ t.q } n(p^*) \subset B_\epsilon, \text{ avec } B_\epsilon \text{ un disque de diamètre } \epsilon \text{ et } n(p^*) = \{p \in l, l \in E / |p^* - p| = \epsilon\}$$

Un ensemble de points extrémité à ϵ près est considéré comme un seul et même point si l'on peut trouver un disque de diamètre ϵ les contenant (voir Figure 3).

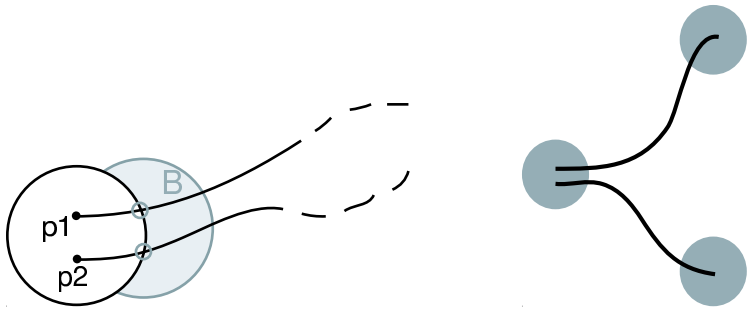

FIG. 3 – **A gauche :** les points p_1 et p_2 sont considérés comme une seule et même extrémité à ϵ près car tous leurs points voisins à une distance de ϵ sont contenus dans B. **A droite :** cet ensemble de lignes contient trois extrémités à ϵ près et ne possède donc pas une structure de ligne à ϵ près.

Une fois ses deux extrémités identifiées, on peut ensuite donner la définition d'un *chemin unique à ϵ près* dans E.

Définition : On considère E^ϵ, la dilatation de E par un disque de diamètre ϵ. E possède un chemin unique à ϵ près ssi E^ϵ ne possède pas de trou et qu'il n'existe pas de disque de diamètre 2ϵ inscrit dans E^ϵ (voir Figure 4).

En effet, si un trou est présent, plusieurs chemins seront possibles, donc l'unicité du chemin ne pourra pas être assurée. Et si le chemin est « trop épais », la nouvelle ligne ne pourra pas couvrir toutes les lignes de l'ensemble initial qu'elle est sensée représenter.

Dans la suite, nous considérons uniquement les partitions où chaque élément possède une structure de ligne à ϵ près.

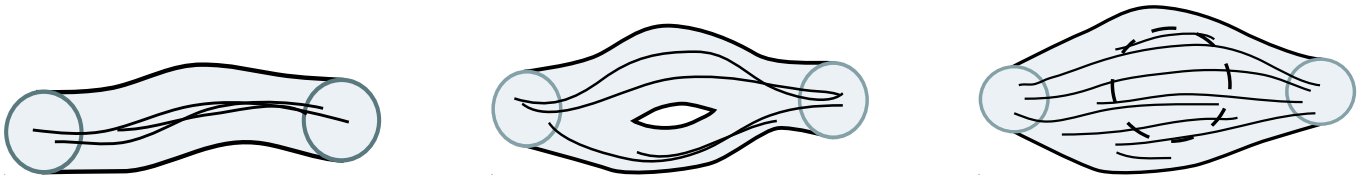

FIG. 4 – L'exemple de **gauche** possède une tructure de ligne à ϵ près : deux extrémités et un chemin unique à ϵ près. Les deux autres exemples ne respectent pas ces contraintes : **au milieu**, la présence d'un trou implique l'existence de plusieurs chemins possibles ; **à droite**, la dilatation des lignes initiales par ϵ est trop épaisse, aucun chemin à ϵ près n'est donc présent.

3 Simplification

Parmis toutes les partitions possibles (respectant la structure de ligne) nous cherchons maintenant une partition offrant un regroupement maximal.

3.1 Partition et mesure d'erreur

On se dote d'une mesure d'erreur m_π et on cherche une partition π^* qui minimise cette erreur :

$$\pi^* = \arg\min_\pi m_\pi$$

Nous commençons par définir l'erreur entre deux lignes puis nous en déduisons la mesure d'erreur d'un groupe de lignes puis celle d'une partition.

Mesure d'erreur d'une paire de lignes L'erreur entre deux lignes correspond à leur distance. Pour la calculer, on cherche les zones où les deux lignes sont « en regard » : les zones de recouvrement.

Définition : une zone de recrouvrement sur une paire de lignes est définie par deux paires de points sur chaque ligne et un chemin unique à ϵ près entre ces deux paires (voir Figure 5).

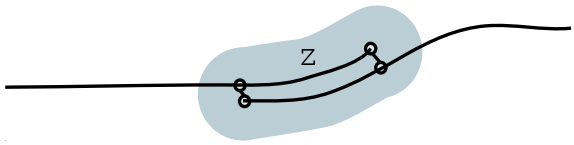

FIG. 5 – Une zone de recouvrement Z entre deux lignes est composée de deux paires de points reliées par un chemin unique à ϵ près. La mesure d'erreur entre ces deux lignes correspond à la distance de Hausdorff sur Z.

On définit la distance entre deux lignes l_i, l_j par la distance de Hausdorff sur la zone de recouvrement Z :

$$m_{i,j} = \max(\min_{i,j}(|x_i - y_j|)) \, , x_i, y_j \in Z$$

Une paire de lignes peut contenir plusieurs zones de recouvrement valides dans les cas où soit elles forment une boucle, soit une des deux lignes se replie et est en regard avec l'autre à plusieurs endroits (voir Figure 6). Il nous faut dans ce cas choisir une de ces zones qui sera celle où les deux lignes seront confondues lors de la simplification. On se contente dans ces cas de conserver uniquement la zone de recouvrement d'erreur minimale ce qui revient à dire que les deux lignes ne seront agglomérées que sur cette zone-là.

Mesure d'erreur d'un groupe de lignes On définit ensuite l'erreur sur un groupe de lignes E par son épaisseur maximale (voir Figure 7) :

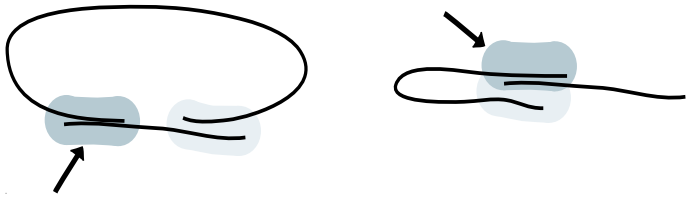

FIG. 6 – Dans un cas particulier comme celui d'une boucle (**à gauche**) ou celui d'un repliement (**à droite**), on conserve uniquement la zone de recouvrement d'erreur minimale.

$$m_E = \max_{i,j \in E} m_{i,j}$$

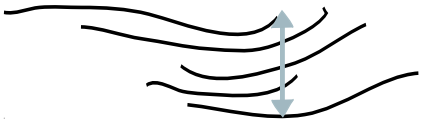

FIG. 7 – L'erreur associée à un groupe de lignes (son épaisseur) correspond à la distance maximale entre deux lignes (les « plus éloignées »).

Mesure d'erreur d'une partition Afin de rendre compte de l'accumulation de l'erreur dans chaque groupe de lignes, on définit la mesure d'erreur d'une partition π comme la somme des erreurs de chacune de ses parties :

$$m_\pi = \sum_{E \in \pi} m_E$$

3.2 Graphe d'agglomération

Une fois les zones de recouvrement détectées, on cherche une partition d'erreur minimale. On utilise pour cela un algorithme glouton : la zone de recouvrement d'erreur minimale est considérée ; puis on agglomère la paire de lignes correspondante et on met à jour les zones de recouvrement incidentes. Cela revient à appliquer un algorithme de contraction d'arêtes sur un graphe où les lignes de L sont les noeuds et où les zones de recouvrement sont les arêtes voir figure 8.

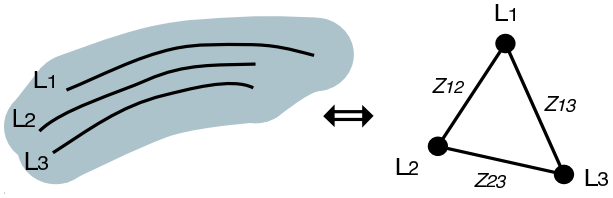

FIG. 8 – L'ensemble de lignes initial est représenté par un graphe où les lignes sont les nœuds et les zones de recouvrement sont les arêtes. La partition est obtenue en utilisant un algorithme de contraction d'arêtes jusqu'à ce qu'il n'y ait plus aucune arête dans le graphe.

On appelle cette structure un graphe d'agglomération. A chaque contraction d'arête (ou agglomération), on effectue les opérations suivantes :

1. Création d'un nouveau noeud qui représente les lignes agglomérées ;
2. Suppression des noeuds et arêtes dégénérées : l'arête contractée, les noeuds de l'arête contractée, mais aussi toute arête incidente à l'un de ces deux noeuds ;
3. Création des arêtes incidentes au nouveau noeud.

L'algorithme s'arrête lorsque toutes les arêtes ont été contractées.

La convergence d'un tel algorithme est garantie, mais uniquement vers un minimum local. En pratique, on obtient des résultats satisfaisants.

4 Abstraction

Une fois l'étape de simplification effectuée, il nous reste de nombreux degrés de liberté pour la génération des nouvelles lignes. Aucune information de l'ensemble initial de lignes n'a été perdue. Différentes approches sont ainsi possibles pour créer ces nouvelles lignes à partir des groupes de π^*.

- On peut décider d'agglomérer progressivement les lignes : chaque paire de lignes donne une nouvelle ligne à chaque étape de la simplification (sans aucun impact sur le déroulement de la contraction). Cette approche peut être utile dans les cas où il est plus intuitif de raisonner par paire de lignes.
- On peut au contraire décider d'agglomérer les lignes en post-traitement : ici un groupe complet de lignes est remplacé par une nouvelle ligne selon une stratégie donnée. Cette approche est préférable lorsque les critères de création portent sur la nouvelle ligne à créer, avec comme seule contrainte que cette ligne soit à une distance inférieure à ϵ de son groupe.
- On peut décider d'interpoler les lignes initiales pour créer une ligne qui passe au mieux par celles-ci ; ou au contraire, choisir de privilégier certaines lignes en donnant un ordre d'importance : la ligne générée suivra les lignes les plus importantes.
- On peut également lisser la ligne résultante, notament au niveau des transitions entre deux lignes à agglomérer.
- Enfin, si l'algorithme est destiné à être appliqué de manière itérative, on peut soit conserver l'ensemble des lignes initiales et à chaque nouvel ajout de ligne, enrichir cet ensemble puis agglomérer ; ou bien on peut, à chaque séquence simplification+abstraction, utiliser les lignes nouvellement créées comme ensemble initial pour l'étape suivante .

Applications

Les choix à faire parmi ces degrés de liberté dépendent de l'application visée. Nous avons implémenté deux applications simples : un outil de tracé progressif et un opérateur de réduction de densité sur un dessin.

Pour l'outil de tracé progressif, nous avons choisi une stratégie d'agglomération progressive où l'on associe à chaque trait une priorité dépendant de son ordre d'apparition : les traits dessinés les plus récemment priment sur les traits anciens. Ainsi, lorsque l'utilisateur dessine un nouveau trait, il peut corriger une portion du dessin car c'est ce dernier trait qui prime sur les autres. De plus, à chaque étape, on réinitialise le processus en repartant des nouvelles lignes agglomérées. L'utilisateur peut ainsi modifier radicalement la forme d'un trait car il n'est pas contraint à rester distant de moins de ϵ du trait initial.

Pour l'opérateur de réduction de densité, nous avons décidé de ne donner aucune priorité aux traits : lorsque l'on décide d'agglomérer une paire de lignes, on utilise une stratégie dite d'interpolation. C'est-à-dire que la ligne résultante est celle qui passe « entre » les deux lignes initiales. Nous avons utilisé pour cela une stratégie progressive, plus simple à programmer. Mais nous aurions pu également utiliser une stratégie en post-traitement, avec des résultats sensiblement identiques.

5 Implémentation

Les lignes en entrée de notre algorithme peuvent être de nature quelconque : on doit seulement pouvoir les échantillonner de manière régulière. Notre système travaille ensuite sur ces lignes échantillonnées.

On utillise une grille d'accélération de taille de cellule ϵ pour le calcul des zones de recouvrement. En partant de l'observation qu'une zone de recouvrement valide contient au moins une extrémité de la paire de lignes initiales, on procède comme suit :

- Pour chaque extrémité e de chaque ligne l_i, on cherche tous les points à une distance inférieure à ϵ de e : ses voisins. Si une ligne l_j porte un de ces voisins, alors on dit que l_i et l_j sont connectées ;

– On extrait ensuite entre chaque paire de lignes ainsi connectées un chemin continu (tel qu'il n'existe pas de point à une distance supérieure à ϵ des autres) ; on obtient ainsi une zone de recouvrement ;
– Enfin, on valide cette zone en comptant les extrémités à ϵ près de la paire de lignes.

On stocke dans cette zone de recouvrement son erreur calculée, ainsi que ses indices de départ et de fin sur chacune des deux lignes.

Puis on construit le graphe. Chaque noeud porte une référence sur sa ligne initiale, chaque arête porte la zone de recouvrement correspondante.

A chaque étape de contraction, l'arête d'erreur minimale est contractée : on crée alors un proxy. Un proxy est une ligne interpolée entre les deux lignes de la paire initiale et qui stocke en chaque point l'erreur (ou épaisseur) maximale en ce point (voir Figure 9). Les lignes initiales sont considérées comme des proxy d'épaisseur 0.

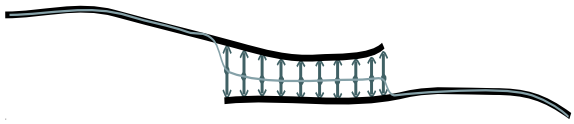

FIG. 9 – Un proxy est créé à chaque étape d'agglomération. C'est une ligne qui représente deux autres lignes et stocke l'erreur entre ces deux lignes sous la forme d'une epaisseur

La création de nouvelles arêtes revient à détecter les zones de recouvrement entre ce nouveau proxy et les autres noeuds du graphe. On ajoute alors l'épaisseur du proxy dans tous les calculs de distance afin de refléter le fait que ce dernier représente un groupe de lignes.

Un nouveau nœud correspondant au regroupement de deux proxy crée une ligne agglomérée selon la stratégie choisie (interpolation ou ordre d'insertion de la ligne). Les lignes résultantes sont les lignes agglomérées présentes dans chaque noeud à la fin de la simplification.

6 Résultats

On commence par donner nos résultats sur des ensembles restreints de lignes afin d'illustrer au mieux les caractéristiques de notre approche. La valeur d'ϵ est indiquée au dessus de chaque figure par un segment.

Dans la Figure 10, on observe le résultat d'un ensemble de lignes trop épais pour pouvoir être représenté par une seule ligne : notre algorithme dans ces cas va générer le moins de lignes possible pour couvrir l'ensemble initial.

Dans la Figure 11, on montre un ensemble de lignes qui respecte toutes les contraintes d'une structure de ligne et la nouvelle ligne qui lui est associée.

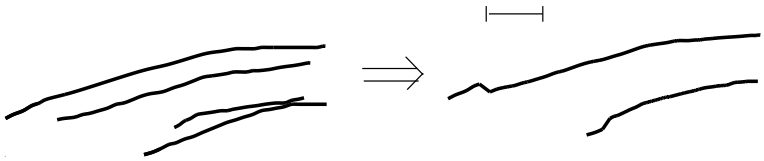

FIG. 10 – Ce groupe de ligne ne peut être représenté par une seule nouvelle ligne car il est trop épais. Deux lignes sont nécessaires.

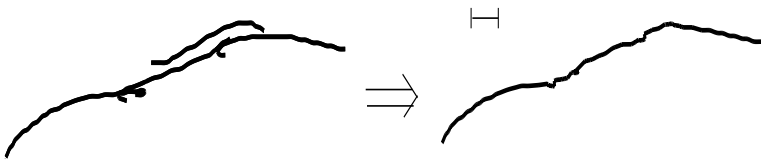

FIG. 11 – Ici, toutes les lignes initiales peuvent être représentées par une unique ligne.

Enfin, nous montrons quelques images de nos deux applications : la Figure 12 montre un ensemble de lignes avant et après mise à l'échelle ; tandis que la Figure 13 montre une séquence de tracé progressif.

Les temps de calcul pour une mise à l'échelle varient selon le nombre de lignes en entrée. Ils sont de l'ordre de quelques secondes. Toutefois, de nombreuses optimisations sont envisageables. Quant à l'outil de tracé progressif, le temps de réaction est instantané puisque seulement quelques lignes (pas plus d'une dizaine en règle générale) sont agglomérées à chaque étape.

Ces résultats correspondent au problème que nous nous sommes posé. Toutefois, ces applications n'exploitent qu'une très faible partie des possibilités offertes par notre système, comme nous allons le voir dans la section suivante.

FIG. 12 – Un ensemble de lignes initial (**à gauche**) et le même ensemble de lignes simplifié à deux échelles différentes (**au milieu et à droite**).

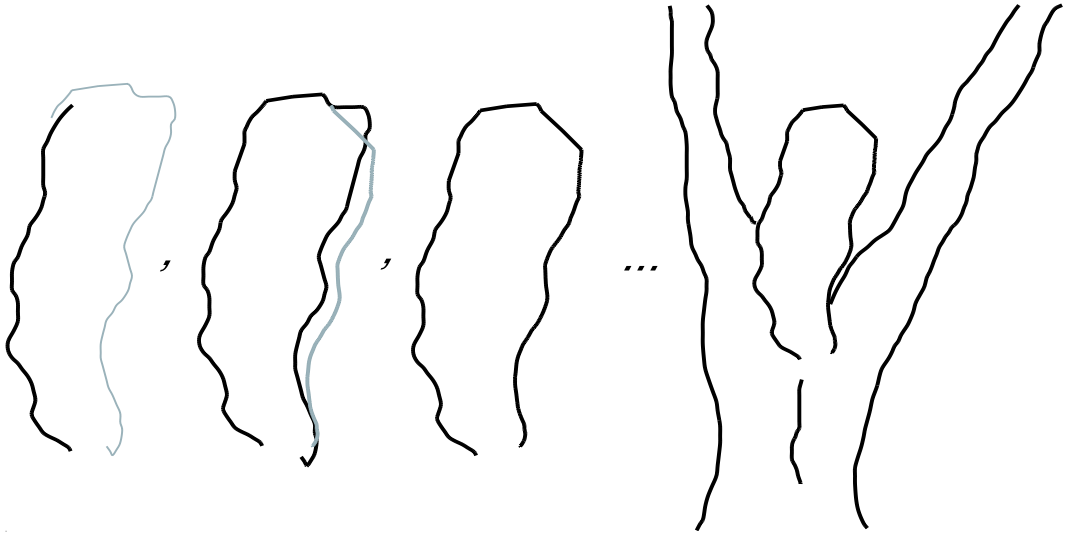

FIG. 13 – Une séquence d'utilisation de l'outil de tracé progressif ; le nouveau trait est affiché en gris, les anciens traits en noir. De **gauche** à **droite** : l'utilisateur étend tout d'abord une ligne existante ; puis il corrige le dessin en ajoutant une nouvelle ligne possédant la forme désirée ; ainsi de suite jusqu'à obtenir le dessin de droite.

7 Conclusion et perspectives

Nous avons présenté un cadre de travail général pour les méthodes de simplification de lignes. Au coeur de notre approche réside la volonté de conserver la structure du dessin initial sans toutefois trop contraindre le résultat : l'utilisateur dispose d'assez de degrés de liberté pour pouvoir d'adapter le système à diverses applications. De plus, la nature et la forme des lignes en entrée peuvent être quelconques.

Nous avons présenté deux applications simples : un outil de tracé progressif et une méthode de mise à l'échelle d'un dessin.

Dans le futur, nous allons tout d'abord enrichir ces deux applications : ajouter des fonctionnalités d'édition à l'outil de tracé progressif comme la génération de lignes droites, l'aimantation, la création de boucles ; intégrer

des informations supplémentaires dans la méthode de mise à l'échelle, comme des données de style (respecter la forme, la couleur des lignes initiales) ou des données 3D (profondeur, identifiant provenant d'un rendu 3D). Nous allons également aborder une nouvelle application : la création de transitions pour les niveaux de détail. Ici l'utilisateur spécifiera deux niveaux successifs et notre système déterminera une transition d'un niveau à l'autre.

Afin de rendre cela possible, nous aimerions ajouter de nouvelles fonctionnalités à notre système, notament : permettre la définition d'une mesure d'erreur incorporant des critères supplémentaires (couleur, identifiant, profondeur 3D, etc) ; créer des stratégies d'agglomération en post-traitement ; ou tenir compte des cas spéciaux comme les boucles ou les repliements (Figure 6).

Nous souhaitons également caractériser la qualité de la solution obtenue par notre algorithme glouton : dans quels cas converge-t-on vers un optimum global et quelle est la différence entre un minimum local et un optimum global ?

Remerciements

Merci à Xavier Decoret, Laurent Favreau, Stephane Grabli, John Hughes, Aurelien Martinet, Sylvain Paris et Cyril Soler pour leurs avis et conseils. Merci à Gilles Debunne pour QGLViewer ([Deb]).

Références

[Deb] Gilles Debunne. Qglviewer : http ://artis.imag.fr/members/gilles.debunne/qglviewer/index.html.

[DS00] Oliver Deussen and Thomas Strothotte. Pen-and-ink illustration of trees. *Proc. SIGGRAPH*, 2000.

[GDS04] Stéphane Grabli, Frédo Durand, and François Sillion. Density measure for line-drawing simplification. In *Proceedings of Pacific Graphics*, 2004.

[GG01] Gooch and Gooch. *Non-Photorealistic Rendering*. AK-Peters, 2001.

[IMKT97] Takeo Igarashi, Satoshi Matsuoka, Sachiko Kawachiya, and Hidehiko Tanaka. Interactive beautification : A technique for rapid geometric design. In *UIST (ACM Annual Symposium on User Interface Software and Technology)*, pages 105–114, 1997.

[KMM+02] Robert D. Kalnins, Lee Markosian, Barbara J. Meier, Michael A. Kowalski, Joseph C. Lee, Philip L. Davidson, Matthew Webb, John F. Hughes, and Adam Finkelstein. WYSIWYG NPR : drawing strokes directly on 3d models. In *SIGGRAPH 2002*, 2002.

[PHWF01] Emil Praun, Hugues Hoppe, Matthew Webb, and Adam Finkelstein. Real-time hatching. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 579–584, 2001.

[SALS96] Mike Salisbury, Corin Anderson, Dani Lischinski, and David H. Salesin. Scale-dependent reproduction of pen-and-ink illustrations. *Computer Graphics*, 30(Annual Conference Series) :461–468, 1996.

[sbm04] *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, 2004.

[SS02] Thomas Strothotte and Stefan Schlechtweg. *Non-photorealistic computer graphics : modeling, rendering, and animation*. Morgan Kaufmann, San Francisco, CA, USA, 2002.

[WM04] Brett Wilson and Kwan-Liu Ma. Representing complexity in computer-generated pen-and-ink illustrations. In *NPAR*, 2004.

[WPFH02] Matthew Webb, Emil Praun, Adam Finkelstein, and Hughes Hoppe. Fine tone control in hardware hatching. In *Proceedings of NPAR 2002, International Symposium on Non Photorealistic Animation and Rendering (Annecy, France, June2002)*, 2002.

[WS94] Georges Winkenbach and David Salesin. Computer-generated pen-and-ink illustration. *Proc. SIGGRAPH*, 1994.

[ZISS04] Johannes Zander, Tobias Isenberg, Stefan Schlechtweg, and Thomas Strothotte. High quality hatching. *Computer Graphics Forum*, 23(3) :421–421, 2004.

[ZT98] Djemel Ziou and Salvatore Tabbone. Edge detection techniques - an overview. *International Journal of Pattern Recognition and Image Analysis*, 8 :537–559, 1998.