Université de Liège
Faculté des Sciences Appliquées
Département d'Électricité, Électronique et Informatique
Institut Montéfiore

Algorithme de tracé réaliste pour environnement de dessin virtuel

Travail de fin d'études présenté par
Jeunejean Aurélie
pour l'obtention du grade
d'ingénieur civil informaticien

Année académique 2003-2004

Sommaire

Remerciements

Avant toute chose, je tiens à remercier les personnes qui m'ont aidée, de près ou de loin, à mener ce projet à bien.

Je remercie particulièrement Monsieur le Professeur Pierre Leclercq, ainsi que Messieurs Roland Juchmes et Sleiman Azar, tous trois membres du groupe LUCID, pour leur disponibilité et leurs conseils tout au long de mon travail.

Ma reconnaissance également à mon promoteur, Monsieur le Professeur Jacques Verly pour ses avis éclairés.

Tous mes remerciements à Messieurs les Professeurs Yves Lion et Serge Cescotto ainsi qu'à Monsieur Jérôme Tchoufang pour le temps qu'ils ont bien voulu me consacrer.

Enfin, je remercie tous ceux qui ont pris et prendront la peine de lire cet ouvrage.

Introduction

Ce travail de fin d'études est réalisé en collaboration avec *LEMA*, laboratoire d'études méthodologiques architecturales, un département de recherches associé à l'École d'Architecture de la Faculté des Sciences Appliquées de l'Université de Liège. Les activités de ce département sont principalement orientées vers des méthodes assistées par ordinateur pour la conception architecturale et urbaine.

Au sein de *LEMA* s'est constitué le groupe LUCID, *Lab for User Coginition and Intelligent Design*, qui étudie l'ingénierie de conception, particulièrement dans les domaines de l'architecture et de la mécanique. Cette équipe développe[1] *EsQUIsE*, un prototype sur ordinateur expérimental servant à capturer et interpréter le croquis de l'architecte.

Les seuls outils nécessaires à un concepteur pour esquisser un projet étant un crayon et une feuille de papier, le bureau virtuel intégré au système *EsQUIsE* se devait de conserver le caractère naturel d'une telle démarche, et donc d'être le plus intuitif possible. Ainsi, la tablette graphique a des airs de table tout à fait quelconque au format A0. Deux projecteurs et un ordinateur placés au dessus de cette table permettent d'y afficher l'interface *EsQUIsE* avec laquelle l'utilisateur peut interagir à l'aide du seul stylet électronique.

Si le titre de ce projet comporte les mots « tracé réaliste » , il serait plus correct de parler de rendu non-photoréaliste. En effet, bien qu'il ne soit pas fréquent de donner la définition d'un terme par opposition, le rendu non-photoréaliste, *Non-photorealistic rendering*, est bel et bien défini par opposition au rendu photoréaliste, c'est-à-dire réalisme comparable à celui d'une photographie. L'objectif de ce projet est donc d'obtenir, en temps réel, le rendu d'un trait de crayon, d'un surligneur ou d'un pinceau sur ordinateur en se servant des seules informations renvoyées par le système via le stylet électronique, à savoir sa position (X, Y), son inclinaison (α, β), la pression exercée P et une information de temps.

La première section de ce document est consacré à l'état de l'art de quelques unes des techniques de rendu non-photoréaliste. De la section 2 à 4, les trois grands axes de ce projet sont tour à tour détaillés : la modélisation d'une feuille de dessin virtuelle, celle d'un crayon graphite et l'interaction entre ces deux outils. La section 5 présente le prototype développé ainsi que quelques résultats obtenus. Finalement, la dernière section évoque les perspectives d'avenir de ce projet.

[1] sur *Macintosh* et *Windows NT* en *Common LISP*

1 État de l'art

Depuis quelques années, l'infographie s'intéresse de plus en plus aux techniques de rendu non-photoréaliste (NPR) directement ou indirectement inspirées par les arts graphiques traditionnels. Un nombre impressionnant d'articles traite du sujet dans la littérature, et il est donc difficile d'être exhaustif. Cependant, Craig Reynolds a conçu un site web [24] référençant l'ensemble des activités NPR menées aux quatre coins du monde. C'est notamment grâce à cette base de données qu'il m'a été possible d'établir l'état de l'art des techniques NPR qui nous intéressent le plus, à savoir les dessins au trait et au crayon ainsi que l'aquarelle. Cette section synthétise donc les principaux articles publiés pour chacune de ces techniques, ainsi qu'un logiciel spécialement conçu pour les architectes utilisant certains procédés de rendu NPR.

Dessin au trait

En 1994, Georges Winkenbach et David Salesin publient un article [29] qui décrit un système permettant de produire automatiquement des dessins au trait, à partir d'une scène 3D. Le système est basé sur la technique dite de texture au trait. Des textures cohérentes entre elles et représentant différents niveaux d'intensité sont produites séparément. Le système supporte les variations dues à la transformation de perspective et à l'effet d'orientation de l'objet rendu au trait. Une attention toute particulière a été portée sur le fait que le rendu au trait automatique soit aussi naturel que le dessin au trait classique. Par exemple, un traitement spécial permet d'éviter l'aspect monotone et impersonnel d'une image où tout espace est rempli de traits (voir figure 1).

FIG. 1 – Dessin au trait - Georges Winkenbach et David Salesin [29]

Un autre article [25] de Michael Salisbury et al. explore l'aspect interactif du dessin au trait. Le système agit comme un outil de composition classique, à une différence notable près : le remplissage des surfaces s'effectue selon une texture de trait pré-déterminée et stockée dans une librairie. Le système supporte plusieurs sophistications, comme par exemple un traitement spécial sur le bord des surfaces plates, afin d'éviter qu'elles n'apparaissent trop bien coupées (voir figure 2).

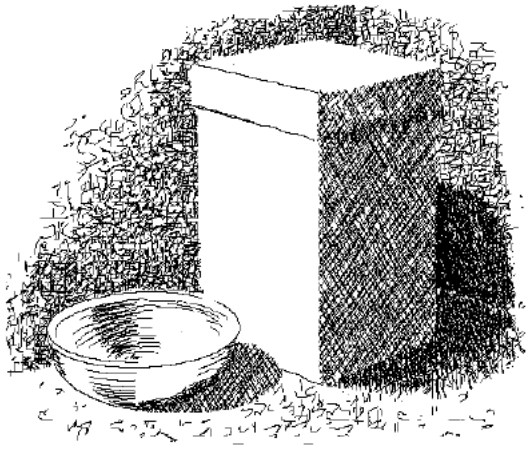


FIG. 2 – Dessin au trait - Michael Salisbury et al. [25]

Plus récemment, Emil Praun et al. ont développé une technique permettant de produire le rendu au trait en temps réel sur base d'un modèle 3D (voir figure 3). La technique décrite dans [22] exploite les possibilités offertes par les cartes graphiques modernes, notamment les projections de textures multiples et les interpolations entre plusieurs niveaux de textures implémentées au niveau du matériel (*MIP map*).

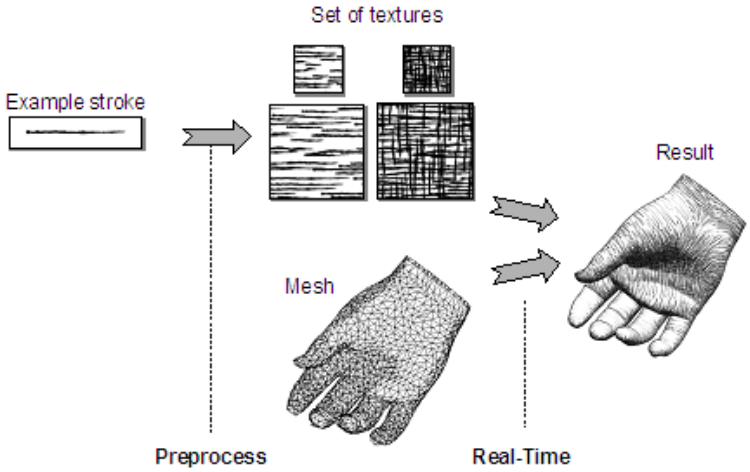


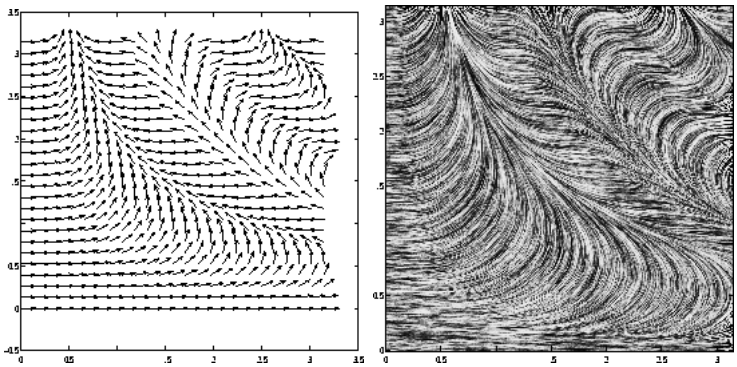
FIG. 3 – Dessin au trait - Emil Praun et al. [22]

Dessin au crayon

Mario Costa Sousa et John Buchanan introduisent dans [26] un modèle qui permet de simuler le trait de crayon. Ils qualifient ce modèle d'observable, car il se base sur les observations microscopiques et macroscopiques des interactions entre les différents types de crayons et de papiers. De plus, le système décrit dans cet article permet de simuler les effets des tortillons, estompes et autres gommes (voir figure 4).

FIG. 4 – Dessin au crayon - Mario Costa Sousa et John Buchanan [26]

Dernièrement, Xiaoyang Mao et al. présentent dans [15] une méthode basée sur la représentation de champs de vecteurs par des lignes de champs, introduite par Brian Cabral et Leith Leedom [2] et connue sous le nom de *Line Integral Convolution*. Cette méthode a pour but de générer une image représentant les lignes tangentes au champ de vecteurs en déformant une image de texture pseudo-aléatoire.



(a) Champ de vecteurs (b) Texture déformée

FIG. 5 – *Line Integral Convolution*

Aquarelle

Un article impressionnant de Cassidy Curtis et al. introduit une simulation informatique de la technique de dessin à l'aquarelle [4]. Les auteurs se fixent un objectif difficile qu'ils parviennent pourtant à atteindre : produire des dessins indiscernables de vraies aquarelles. Pour commencer, ils définissent les propriétés intrinsèques de l'aquarelle et établissent le catalogue des techniques de base utilisées par les aquarellistes traditionnels. Afin de simuler ces effets, les auteurs se servent des équations de Navier-Stokes pour modéliser le mouvement des fluides qui transportent les pigments. L'étape suivante consiste à calculer le dépôt des particules de pigments sur la surface rugueuse du papier. L'eau est par la suite absorbée par la pression capillaire du papier, en modifiant la distribution du colorant. Le modèle composite de Kubelka-Munk permet de calculer l'effet de coloration dû aux diverses couches superposées. Ainsi, l'article décrit la simulation de divers effets importants, par exemple l'effet du pinceau sec, l'effet d'assombrissement des bords, l'effet de granulation, l'effet de retour d'eau (voir figure 6 [2]). Bien que le système soit numériquement stable, la technique présentée est très coûteuse en terme de volume de calcul - une image de taille modérée pouvant demander plusieurs heures de calcul.

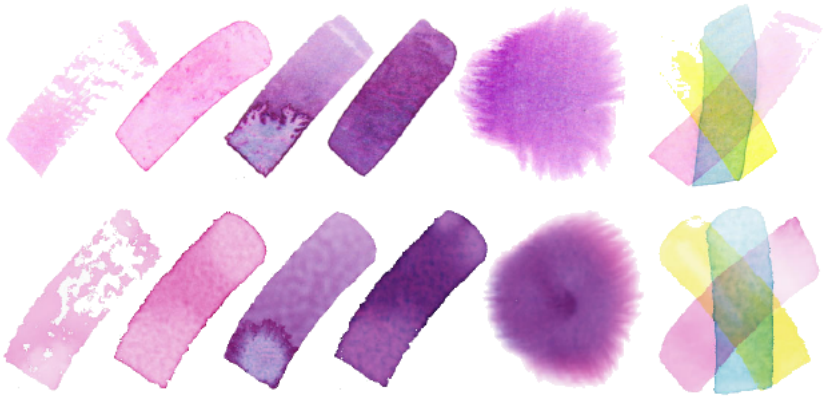


FIG. 6 – Aquarelle - Cassidy Curtis et al. [4]

[2] Les vraies aquarelles se trouvent sur la ligne du dessus !

Piranesi

Piranesi est un système de dessins architecturaux, expérimental à l'origine, et devenu commercial par la suite. Il est interfacé avec la plupart des logiciels de modelage géométrique, afin d'obtenir la géométrie de base. À partir de celle-ci, le système permet de dessiner en perspective, comme si on peignait directement sur les surfaces disposées dans l'espace à trois dimensions. Le système emploie un modèle du monde tridimensionnel projeté sur une sphère unitaire qui entoure le point d'où la scène est observée. À l'aide des outils interactifs, l'utilisateur de ce système est capable de superposer des éléments (croquis et dessins de différentes provenances), de les ajuster, de les mettre en perspective, et de dessiner par-dessus comme s'il s'agissait de feuilles de dessin disposées dans un espace virtuellement créé à partir de ces éléments disparates.

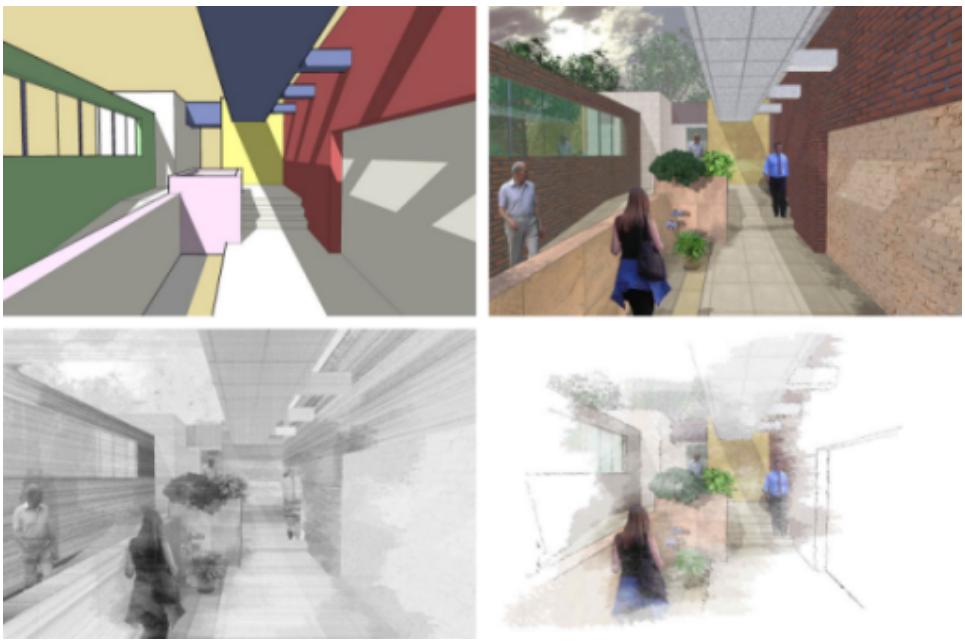


FIG. 7 – Piranesi - Illustration des différentes étapes du rendu

Conclusion

La grande majorité des techniques de rendu non-photoréaliste rencontrées se base sur une amélioration d'une modélisation en trois dimensions déjà existante. Même si les résultats obtenus sont parfois très impressionnants, ces techniques ne conviennent pas du tout à notre projet, qui se doit de conserver un caractère interactif, tout en restant temps réel. Considérant ces deux aspects, seuls les articles de Mario Sousa et Cassidy Curtis sont intéressants et constituent les références principales de ce projet.

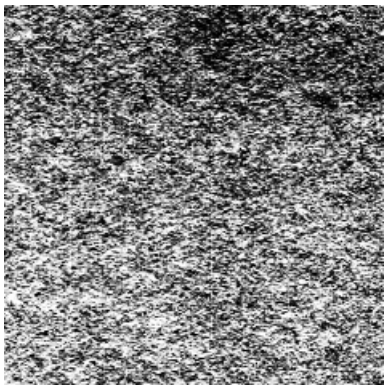
2 La feuille de dessin

La grande diversité des supports de dessin sur le marché n'est pas le fruit du hasard. En effet, chaque type de feuille présente des caractéristiques qui influencent fortement le rendu d'un croquis. Par exemple, plus une feuille est rugueuse, moins les traits du dessinateur seront uniformes. D'autres paramètres comme le grammage, l'épair ou le processus de fabrication différencient encore davantage les feuilles de papier. Or, dans la littérature, bon nombre d'articles traitant de rendu réaliste ne s'intéressent pas à la modélisation de la feuille étant donné que le rendu est produit a posteriori. Par contre le caractère temps réel de ce projet implique la gestion de l'interaction entre le crayon et la feuille et donc la modélisation de cette dernière. De la qualité de cette modélisation dépend donc le rendu final du dessin virtuel. C'est pourquoi cette étape a constitué la majeure partie du travail présenté ici.

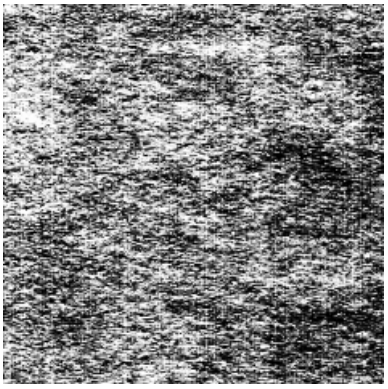
Une feuille de papier est fabriquée à partir d'une pâte de fibres végétales ou synthétiques étalée en couches minces. Beaucoup d'éléments non fibreux entrent également dans la composition du papier : glaise, carbonate de calcium, résine, colorants, ... Dès lors, observée au microscope, la feuille révèle une succession de bosses et de creux. Bien que cet entrelacement de fibres donne à la feuille ses caractéristiques, c'est un modèle plus simple formé d'un champ de hauteurs qui sera utilisé ici pour la modéliser en trois dimensions.

L'œil humain différencie les feuilles selon l'aspect qu'elles présentent, ce qui permet de voir le relief microscopique d'une feuille de papier comme une texture. La génération d'une feuille virtuelle peut donc se résumer à la synthèse d'une texture (voir section 2.2 à 2.4). Mais avant toute chose, l'acquisition d'un modèle de chaque feuille utilisée par les architectes est nécessaire. La section suivante présente les quatre techniques envisagées : la numérisation, la microscopie électronique, la microscopie optique et l'interférométrie de speckle.

(a) Canson C à grains - 125 g/m^2



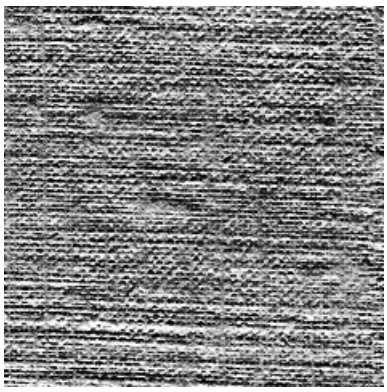
(b) Canson Bristol - $180g/m^2$



(c) Canson Calque satin - $92g/m^2$

(d) Canson Mi-teinte ocre - $160g/m^2$

(e) Fabriano WaterColor - $300g/m^2$



(f) ClaireFontaine (huile) - $240g/m^2$

Résolution	200 dpi	Zones claires	255
Type de sortie	256 niveaux de gris	Ombre	200
Echelle de sortie	100%	Demi-teintes	1.0
Niveau de netteté	extrême	Inversion des couleurs	non

FIG. 8 – Images obtenues par numérisation

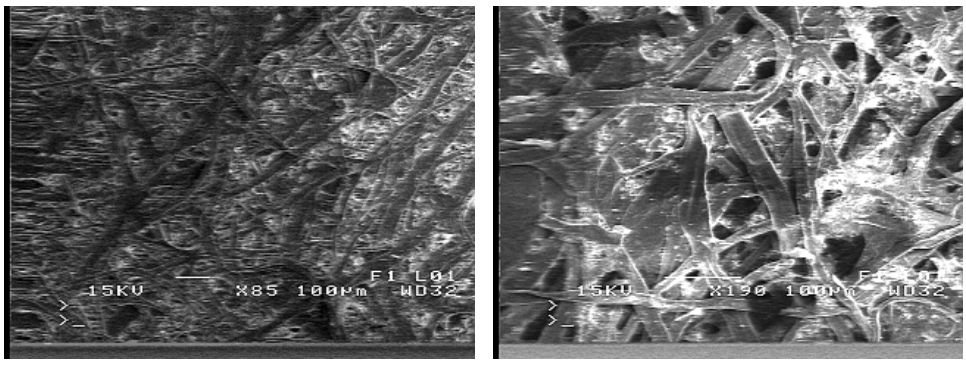
2.1 Acquisition d'un modèle

La numérisation

La première idée qui vient à l'esprit lorsque l'on veut obtenir une image d'un document est d'utiliser un scanner : la numérisation d'une feuille de papier permet d'obtenir sans difficulté une image en niveaux de gris. Il est toutefois intéressant de se poser la question de l'origine de ces niveaux. S'agit-il réellement de creux - plus profonds et par conséquent moins éclairés - ou s'agit-il de l'ombre des bosses dues au balayage du scanner ? Il semble que chacune des hypothèses puisse être retenue. En effet, la numérisation d'un même échantillon de papier dans les deux sens de balayage ne présente pas le même aspect, les versants de chacune des bosses étant soit éclairés, soit ombragés. Cependant, l'image en niveaux de gris obtenue présente bien la même apparence que le relief palpable et visible à l'œil nu. Cette méthode constitue donc tout de même une bien meilleure approximation de la texture d'une feuille que la génération d'un champ de hauteurs pseudo-aléatoires. La figure 8 reprend les images obtenues par numérisation des échantillons de papier fournis avec un scanner Hewlett-Packard PSC 1210.

La microscopie électronique de balayage

La microscopie électronique de balayage, *Scanning Electron Microscopy*, est utilisée dans beaucoup d'applications d'acquisition d'images. La plupart d'entre nous a certainement déjà vu la photo d'un CPU ou d'une tête d'une mouche prise au SEM. Le niveau de détail y est étonnant, bien meilleur que celui de n'importe quel système optique. Bien que le Centre Spatial de Liège [3] soit équipé d'un tel appareil, cette méthode n'a pas été retenue car c'est l'utilité même de la microscopie électronique qui est mise en cause. En effet, l'extrême précision n'est obtenue que sur une infime partie de la feuille observée. Dès lors, les images obtenues ne reflètent pas son aspect général. La figure 9 n'est donc présentée qu'à titre informatif.

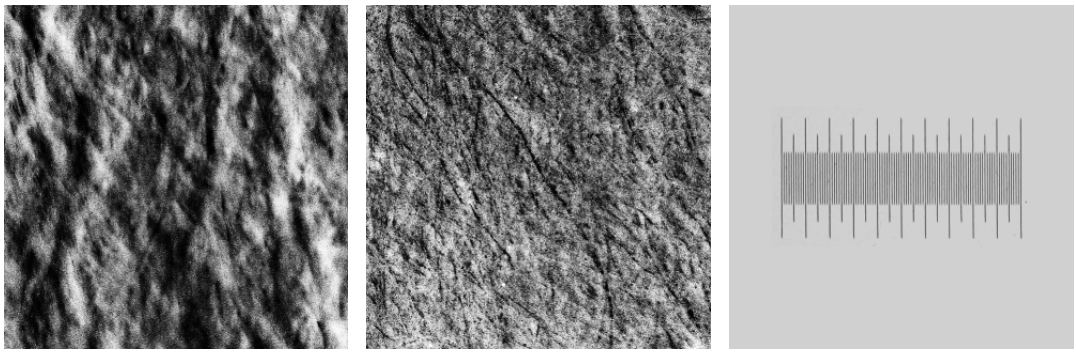


(a) Vue de face (b) Vue en coupe

FIG. 9 – Images d'une feuille de papier obtenues par la microscopie électronique de balayage avec un grossissement de 190× pour la vue de face et de 85× pour la vue en coupe[5]

Le microscopie optique

Contrairement à la résolution de microscopie électronique, celle de la microscopie optique, moins importante, se prête mieux à l'analyse d'une feuille de papier. Non seulement la surface observée est plus grande mais la précision est bien moindre, laissant apparaître l'aspect global de la texture. Les photos de la figure 10 ont été prises avec un microscope optique Olympus BX60M dans le service de Métallurgie & Science des Matériaux.



(a) Canson C à grains® (b) Canson Bristol (c) Micromètre

FIG. 10 – Images obtenues au microscope optique

Interférométrie de speckel

Le speckle est un phénomène d'interférence optique qui peut être observé lorsque des objets sont illuminés par une lumière laser. Cet effet est d'aspect granulaire, il présente des « taches » claires et sombres, causées, respectivement, par les interférences constructives et destructives de la lumière diffusée. La taille apparente de ces grains de speckle dépend de la distance d'observation mais également de l'angle d'inclinaison de la source lumineuse. Le phénomène est en général indésirable, surtout en holographie, mais il présente des propriétés physiques remarquables mises à profit dans l'interférométrie.

L'interférométrie de speckle s'effectue soit par corrélation d'intensité, soit par mesure de phase. Suivant la technique employée, elle permet aussi bien d'analyser des déformations mécaniques que de caractériser la rugosité des surfaces [18] [11]. Suggérée par Monsieur le Professeur Yves Lion du Département de Physique, cette technique n'a pu être expérimentée pour des raisons pratiques. D'une part, l'Université de Liège ne dispose pas d'un dispositif « prêt-à-l'emploi » et d'autre part, les contacts pris avec d'autres universités [3] disposant des instruments de mesure nécessaires n'ont pas abouti. Même si cette approche ne semble pas encore avoir été utilisée pour caractériser la surface du papier à dessin, il serait intéressant d'y consacrer plus de temps dans un projet ultérieur.

[3]Contacts : Ibtissame Malouli, Service d'Optique et Acoustique de l' Université libre de Bruxelles et Pierre Slangen, Centre des Matériaux de Grande Diffusion de l'École des Mines D'Alès.

2.2 Bruit de Perlin

Les générateurs de nombres pseudo-aléatoires sont utilisés dans bon nombre d'applications pour rendre moins probable, donc plus naturel, le mouvement d'objets, l'aspect visuel d'une texture générée, etc. Cette méthode, bien qu'utile, donne cependant des résultats souvent trop artificiels. La nature essentiellement composée de fractales fournit une autre approche du bruit dont l'exemple le plus parlant est le relief d'une montagne qui possède des niveaux de détail de plus en plus petit : montagnes, collines, rochers, cailloux, ... L'herbe, la mer, le mouvement des branches et les dessins du marbre sont autant d'exemples tout aussi connus. Le bruit de Perlin[17] recrée ces variations en additionnant des fonctions de bruit à différentes fréquences et amplitudes (voir figure 11). Deux fonctions élémentaires sont donc nécessaires à la génération du bruit de Perlin : une de bruit et une d'interpolation.

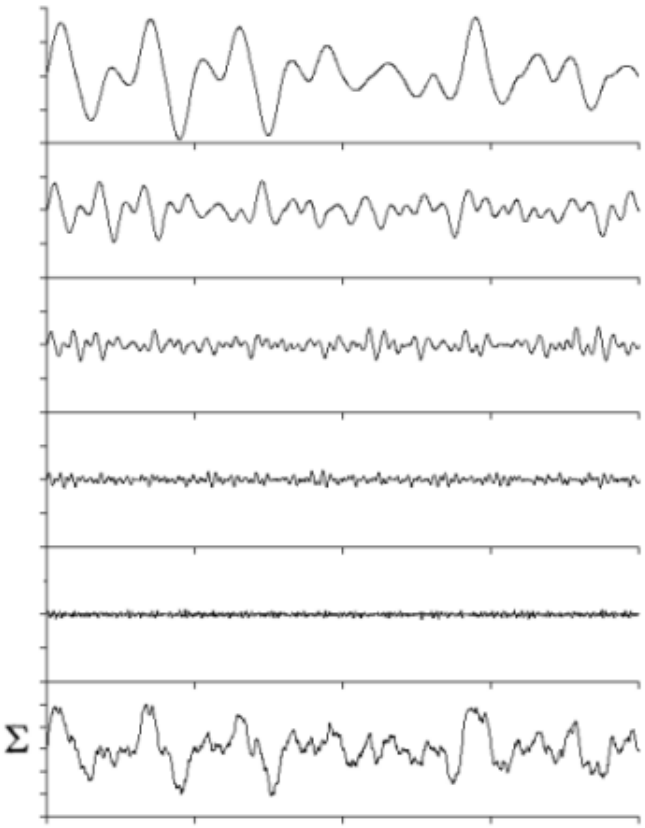


FIG. 11 – Illustration du bruit de Perlin à une dimension. Les cinq premiers graphes sont des fonctions de bruit pour des fréquences et amplitudes différentes et le dernier, obtenu par addition des cinq autres, est le résultat final.

La fonction de bruit doit être un générateur de nombres pseudo-aléatoires à graine, c'est-à-dire une fonction retournant un même résultat pour un même paramètre d'entrée appelé graine. C'est dans cette caractéristique de la fonction de bruit que réside toute la puissance du bruit de Perlin. En effet, les textures obtenues sont des textures dites procédurales c'est-à-dire basées sur des équations mathématiques. Non seulement elles ne demandent aucune ressource mémoire mais il est possible de zoomer à l'infini sans voir apparaître une pixelisation.

Le choix de la fonction d'interpolation est totalement libre. Toutefois, parmi les trois fonctions d'interpolation les plus couramment utilisées, à savoir linéaire, cubique et cosinus, la première est trop grossière et la seconde demande un temps de traitement plus important. L'interpolation cosinus est donc un bon compromis.

L'exemple de la figure 12 illustre le principe à deux dimensions et utilise une fréquence double et une amplitude réduite de moitié pour chaque fonction de bruit successivement ajoutée. Selon les caractéristiques de la texture voulue, d'autres relations fréquences - amplitudes peuvent être choisies. Pour reprendre l'exemple de la montagne, une plaine rocailleuse serait obtenue par des amplitudes faibles aux basses fréquences.

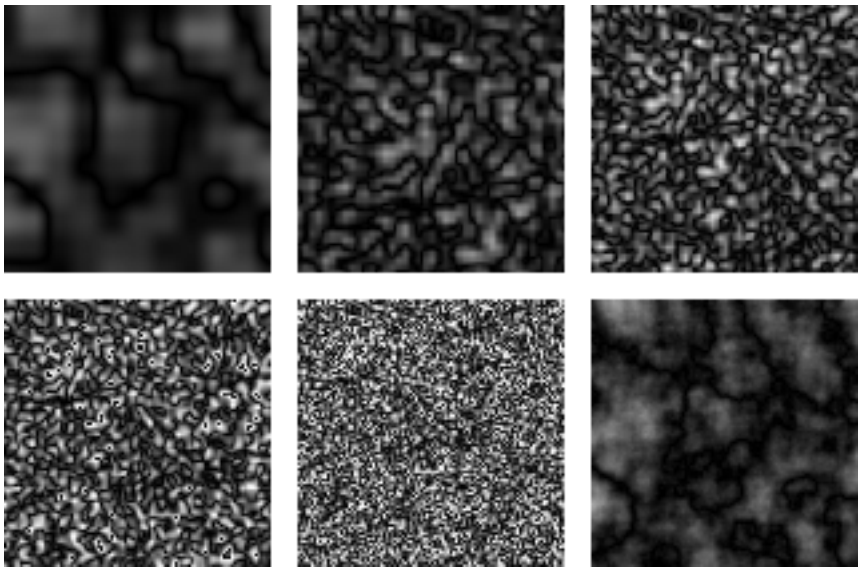


FIG. 12 – Illustration du bruit de Perlin à deux dimensions. Comme dans la figure précédente, la sixième image est le résultat final, obtenue par addition des cinq premières.

Cette méthode laisse entrevoir de larges possibilités, notamment du point vue implémentation, mais aucun résultat probant n'a pu être obtenu jusqu'à présent pour la génération d'une feuille de papier virtuelle.

2.3 *Quilting*

Les techniques dites de *Quilting* synthétisent une nouvelle texture visuellement semblable à un échantillon donné en accolant des morceaux, appelés blocs ou *patches*, issus de cet échantillon. Elles se distinguent dans la façon de choisir les blocs et la manière de les juxtaposer dans l'image finale.

Dans la littérature, bon nombre d'articles traitent du sujet : déterminer précisément quelle est la meilleure forme des blocs pour une texture donnée et comment les assembler le plus harmonieusement possible reste une question ouverte. Tous ces articles font référence à trois méthodes de base que nous allons détailler après avoir évalué la taille optimale du patch pour la texture qui nous concerne.

2.3.1 Taille optimale d'un patch

La forme des blocs dépend bien évidemment de la structure de la texture initiale. Si l'échantillon soit un mur de briques, les blocs seront plutôt rectangulaires. Si par contre l'image représente des cerises, ils seront plutôt carrés. Boris Wang [27] propose un algorithme d'optimisation de la forme des blocs basé sur une étude statistique de l'image de départ.

Soient I l'image initiale de taille $H \times W$ en niveaux de gris, $h_b \times w_b$ la taille d'un bloc et B l'ensemble de toutes les positions possibles du bloc dans l'image, tel que $B_{i,j}$ définit la sous-image de I située en (i, j) de taille $h_b \times w_b$ c'est-à-dire

$$B = \{(x, y) \,|\, x \in [1, H - h_b],\ y \in [1, W - w_b]\}$$

Définissons alors $Stat(h_b, w_b)$ comme le degré de stationnarité du bloc de taille $h_b \times w_b$ dans l'image,

$$Stat(h_b, w_b) = \frac{1}{|B|}.\sigma\{\mu(B_{i,j})\} \qquad (i, j) \in B$$

où $\sigma(*)$ et $\mu(*)$ sont respectivement l'écart type et la moyenne de la série $*$

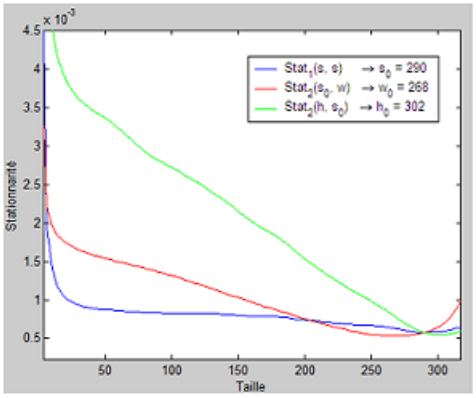
L'algorithme se décompose alors en deux étapes.

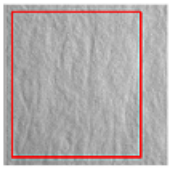
1. le bloc optimal est supposé carré et il faut déterminer le premier minimum local de $Stat(s, s) \quad \forall\, s \in [1, min(H - s, W - s)]$. Supposons qu'il soit en $s = s_0$.
2. déterminer le premier minimum local de $Stat(h, s_0) \quad \forall\, h \in [1, H - h]$ soit $h = h_0$ et de $Stat(s_0, w) \quad \forall\, w \in [1, W - w)]$ soit $w = w_0$.

La taille du bloc optimal est alors donné par $h_0 \times w_0$.

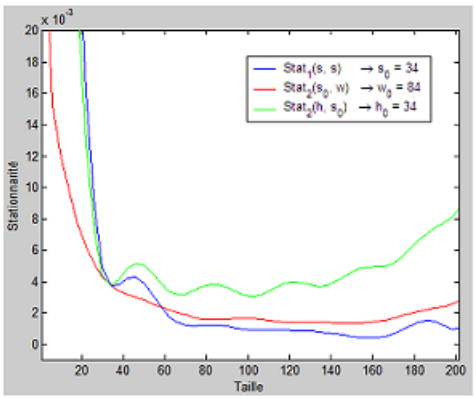
La figure 13(a) illustre les résultats obtenus avec une des photos prises au microscope optique (figure 10(a)). La taille optimale du patch obtenu n'est pas du tout acceptable. En effet, elle est de 302×268 pixels pour une image de 340×340 pixels. Donc, lors du *Quilting* proprement dit, les blocs de l'image finale auront tous une partie commune et

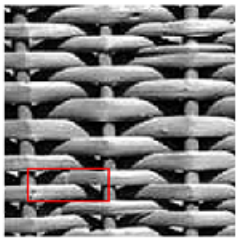
par conséquent des *patterns* apparaîtront. C'est exactement une solution à ce genre de situation que l'algorithme prétendait fournir.





(a) Taille de l'image 340 × 340 pixels - Taille du bloc optimal (en rouge) 302 × 268 pixels





(b) Taille de l'image 128 × 128 pixels - Taille du bloc optimal (en rouge) 34 × 84 pixels

FIG. 13 – Résultats de l'algorithme de recherche du patch optimal

Ce mauvais résultat provient de la nature même de notre texture. Sans être stochastique, elle n'est cependant pas structurée contrairement à celle de la figure 13(b), pour laquelle l'algorithme apporte une très bonne solution. L'article [27] suggère d'améliorer les performances de l'algorithme en égalisant préalablement les niveaux de gris de l'image. Malgré ce pré-traitement, la taille optimale du patch reste, dans le cas qui nous occupe, beaucoup trop grande par rapport à la taille de l'image. En conséquence, des blocs de forme carrée seront utilisés par la suite et leur taille optimale sera obtenue par des tests d'essai-erreur.

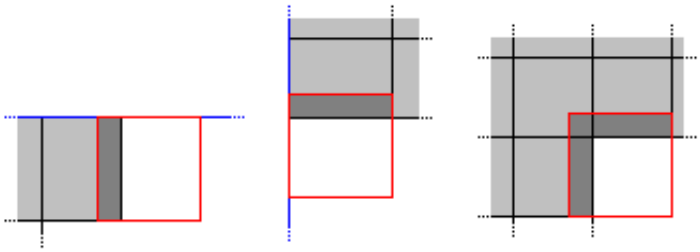
2.3.2 *Quilting* Pseudo-Aléatoire

Le *Quilting* pseudo-aléatoire est de loin la méthode la plus simple et la plus rapide. Inspirée de [30] et [21], elle consiste à choisir de manière pseudo-aléatoire des blocs de taille fixe dans l'image d'entrée et à les recopier dans l'image de sortie.

La figure 16(b) illustre cette technique de base. Mais comme nous l'avons déjà constaté précédemment, la texture d'une feuille de papier ne peut être qualifiée de stochastique. Dès lors, la limite entre les blocs est bien visible. Quelle que soit la taille des blocs et même si un effet de flou est ajouté au niveau de leur frontière, l'aspect du résultat ne paraît pas naturel.

2.3.3 *Quilting* avec Contrainte de Recouvrement

Afin d'améliorer l'aspect de la texture aux frontières des blocs, A. Efros et W. Freeman ont introduit dans [7] la notion de *Quilting* avec contrainte de recouvrement. Un bloc, au lieu d'être choisi pseudo-aléatoirement, est sélectionné pour ses caractéristiques communes avec ses voisins. Une zone de recouvrement est ainsi définie et sert de patron à la sélection du nouveau bloc. Selon l'état d'avancement du processus, le patron présente une des formes reprises à la figure 14.



(a) Cas j = 0 (b) Cas i = 0 (c) Cas général

FIG. 14 – Formes du patron. En bleu : le bord de l'image, en gris clair : les *patches* déjà placés, en gris foncé : le patron, en rouge : l'emplacement du *patch* à trouver.

Algorithme

Soient I l'image initiale de taille $H_I \times W_I$, O l'image de sortie de taille $H_O \times W_O$, B un bloc de taille b et P le patron. L'erreur de recouvrement entre B et P est définie par

$$E(B) = \sum_{i,j} \|O(i,j) - I(i,j)\|^2 \qquad \forall (i,j) \in P$$

Soient o la taille du plus petit côté de la zone de recouvrement et S le seuil d'erreur maximum. Sous les hypothèses,

$$o < b, \qquad b < H_I \quad et \quad b < W_I$$

l'algorithme de Quilting avec contrainte de recouvrement est le suivant :

begin
  $i, j := 0, 0$
  ; *do* $(i + o < H_O) \rightarrow$
    do $(j + o < W_O) \rightarrow$
      if $(i = 0 \quad and \quad j = 0) \rightarrow$
        "Choisir pseudo-aléatoirement un bloc dans I"
      □ $(i \neq 0 \quad or \quad j \neq 0) \rightarrow$
        do ($\forall$ bloc n de taille $b \times b \subset I$) $\rightarrow$ "Calculer la surface d'erreur $E(n)$"
        od
        ; $E_{min} := min(E)$
        ; "Construire la liste L des blocs candidats c.à.d dont $E(n) \leq E_{min} * (1 + S)$"
        ; "Choisir pseudo-aléatoirement un bloc dans L"
      fi
      ; "Copier le bloc choisi dans O"
      ; $j := j + (b - o)$
    od
    ; $i := i + (b - o)$
  od
end

Améliorations des performances

Comme le montre la figure 16(c), cette méthode procure de meilleurs résultats que le *Quilting* pseudo-aléatoire. Si les frontières entre les blocs n'ont pas totalement disparu, elles sont nettement moins visibles. Cependant, le temps de traitement en a bien évidemment pâti et deux modifications ont été apportées à l'algorithme original en vue de l'optimiser. Premièrement, la recherche d'un bloc candidat ne se fait plus par force brute mais selon un facteur d'échelle $\in$ [5..10]. Et deuxièmement, le calcul de l'erreur de recouvrement utilise une norme L_1 au lieu de la norme L_2. Ces deux modifications n'altèrent pas les résultats obtenus avec la version originale mais divisent par cinq au moins le temps de génération de la nouvelle image.

2.3.4 *Quilting* avec Découpe d'erreur minimum

Cette technique également présentée dans [7] doit être vue comme une amélioration de la précédente. En effet, la zone de recouvrement n'est plus simplement occultée par le nouveau bloc choisi mais une découpe y est effectuée au niveau des pixels qui sont les plus similaires c'est-à-dire où l'erreur de recouvrement est minimale (voir figure 15). Il s'agit là du problème de chemin de coût minimum aisément soluble avec l'algorithme de Dijkstra ou en programmation dynamique.

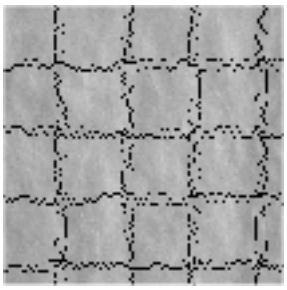


FIG. 15 – Chemin de coût minimum

Soient deux blocs qui se recouvrent en formant un patron vertical de taille $b \times o$. Si B_{o_1} et B_{o_2} désignent les zones de recouvrement respectivement du premier et du second bloc, alors la surface d'erreur est définie par

$$e = |B_{o_1} - B_{o_2}|^2 \tag{2.1}$$

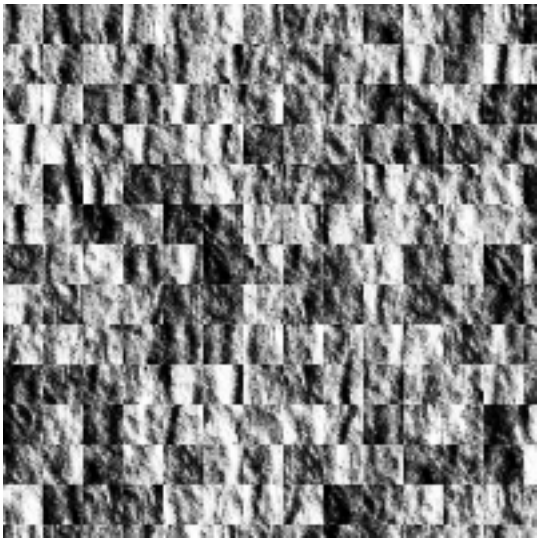
Pour trouver le chemin de coût minimum, la surface d'erreur e est parcourue pour $i = 2 \,..\, b$ et l'erreur minimum cumulative E est calculée pour tous les chemins selon

$$E_{i,j} = e_{i,j} + \mathsf{min}(E_{i-1,j-1},\, E_{i-1,j},\, E_{i-1,j+1}) \qquad j = 1 \,..\, o \tag{2.2}$$

Une fois le parcours de e terminé, la valeur minimale de la dernière ligne de E correspond à la fin du chemin d'erreur minimum vertical et il suffit de remonter E pour trouver la meilleur découpe à faire. Un raisonnement similaire peut être fait pour un patron horizontal.

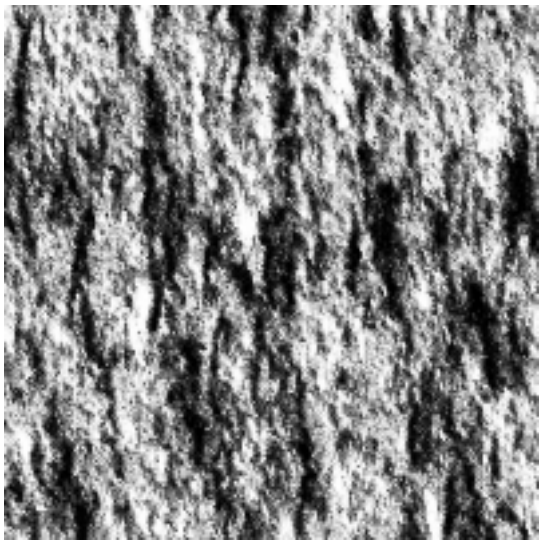
Comme le montre la figure 16(d), cette méthode permet de faire disparaître complètement les frontières rectilignes des blocs. Du point de vue des performances, ce n'est pas tant le calcul de découpe des frontières qui nécessite le plus de temps de traitement, mais bien la recherche d'un bloc candidat. Selon [13], cette recherche peut être accélérée en utilisant la notion d'arbre à k-dimensions, *k-D Tree* ou encore la quantification vectorielle développée à la section 2.4.3.

(a) Image originale



(b) *Quilting* pseudo-aléatoire - 600×600 pixels en 0.01 *seconde*

(c) *Quilting* avec contrainte de recouvrement - 600×600 pixels avec $b = 12$ pixels et $o = 3$ pixels 28 *secondes*



(d) *Quilting* avec découpe d'erreur minimum - 600×600 pixels avec $b = 12$ pixels et $o = 3$ pixels 29 *secondes*

FIG. 16 – Résultats obtenus pour les différentes techniques de *Quilting*

2.4 Algorithme de Wei-Levoy

Afin de permettre au lecteur de bien comprendre le principe de génération de textures proposé par L. Wei et M. Levoy dans [28], l'algorithme de base est décrit en détails à la section 2.4.1. Les sections suivantes ne sont que des améliorations des performances de cet algorithme de base.

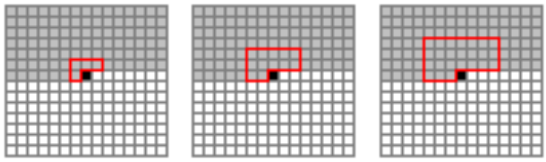
2.4.1 Algorithme de base

Tout comme le Quilting, cette technique nécessite une image d'entrée I considérée comme échantillon. L'image de sortie O est quant à elle initialisée avec du bruit blanc. L'algorithme parcourt alors O pixel par pixel pour lui donner un aspect visuellement semblable à I.

Supposons que le pixel p à remplacer se trouve au milieu de l'image. Les pixels des lignes au-dessus et à gauche ont donc déjà été traités. La valeur à attribuer à ce pixel pour garantir la similarité entre I et O dépend fortement de son voisinage. Soit alors $L(N)$ le voisinage de p en forme de L de taille N (voir figure 17). Les pixels du voisinage de p vont être comparés aux pixels de tous les voisinages de même forme contenus dans I. La nouvelle valeur de p est celle du pixel de l'image d'entrée dont le voisinage est le plus similaire à celui de p (voir figure 18). La norme L_2 est utilisée pour trouver le degré de similarité entre deux voisinages de taille N :

$$distance(L_1, L_2) = \sum_{1 \leq i \leq N} \|p_1(i) - p_2(i)\|_2$$

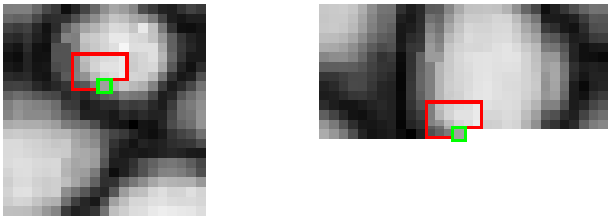
où $p_1(i)$ est le i^{eme} pixel du voisinage L_1.



(a) $L(1)$ (b) $L(2)$ (c) $L(3)$

FIG. 17 – Forme du voisinage $L(N)$ pour $N = 1, 2, 3$. En gris clair, les pixels déjà traités, en noir le pixel actuel et en rouge son voisinage.

Le choix de la forme du voisinage est essentiel. En effet, une forme causale ou asymétrique ou encore en forme de L, c'est-à-dire qui ne contient que les pixels déjà synthétisés, donne de bien meilleurs résultats étant donné que les pixels utilisés sont cohérents. En utilisant une forme non-causale ou symétrique, par exemple un carré, du bruit est forcément ajouté faussant le résultat.



(a) Image initiale (b) Image de sortie

FIG. 18 – Processus de Wei-Levoy. En rouge, le meilleur voisinage L_2.

Algorithme

begin
"Initialiser O avec du bruit blanc"
; *do* ($\forall$ pixel $(i,j) \in O$) $\rightarrow$
L_O := "Voisinage de (i,j)"
; $L_{min}, Couleur := NULL, NULL$
; *do* ($\forall$ voisinage L_I de $(k,l) \subset I$) $\rightarrow$
if ($L_{min} = NULL$) $\rightarrow$
$L_{min} := L_I$; $Couleur := I(k,l)$
$\square$ ($distance(L_O, L_I) < distance(L_O, L_{min})$) $\rightarrow$
$L_{min} := L_O$; $Couleur := I(k,l)$
fi
od
; $O(i,j) := Couleur$
od
end

L'image de sortie est considérée comme toroïdale (voir figure 19). L'aléatoire est introduit dans la synthèse de l'image par l'initialisation. Mais seulement seules les N dernières lignes et colonnes de l'image sont réellement utilisées pour introduire cet aléatoire. En effet, après N lignes traitées, plus aucun pixel du voisinage n'est aléatoire. Même si une toute petite partie de l'initialisation de l'image est utilisée, elle conditionne le résultat final.

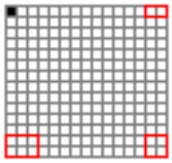


FIG. 19 – L'image de sortie est considérée comme toroïdale.

Les performances de cet algorithme sont assez limitées : la taille du voisinage doit être suffisamment grande pour donner des résultats satisfaisants. En effet, une taille trop petite ne capture que les détails de la texture sans obtenir sa structure générale. Malheureusement, plus la taille du voisinage est élevée, plus le temps de génération augmente. Le problème peut être contourné partiellement en utilisant une représentation en multirésolution de l'image.

2.4.2 Représentation en multirésolution

Partant d'une image de taille $2^N \times 2^N$, la représentation en multirésolution d'une image est donnée par une série d'images de plus en plus petites, appelée pyramide, chacune étant une réduction de la précédente.

Soit I une image de taille $2^N \times 2^N$que l'on veut réduire d'un facteur 2. Cette réduction doit s'accompagner d'un processus de lissage afin d'enlever les hautes fréquences, opération qui s'effectue à l'aide de l'opérateur de convolution suivant :

$$G[I](i,j) = \sum_{m=1}^{M} \sum_{n=1}^{M} w(m,n) * I(2i+m-z, 2j+n-z)$$

où M est la taille du masque de convolution w, z est une constante égale à $\lfloor (M+1)/2 \rfloor$ et où

$$(i,j) \in [0, 2^{N-1}-1] \times [0, 2^{N-1}-1]$$

Le nombre de pixels est réduit de moitié dans chaque dimension, donnant une nouvelle image $G[I]$ quatre fois plus petite que la précédente. L'application itérative du même processus donne une représentation en multirésolution de l'image, c'est-à-dire la suite d'images de tailles décroissantes :

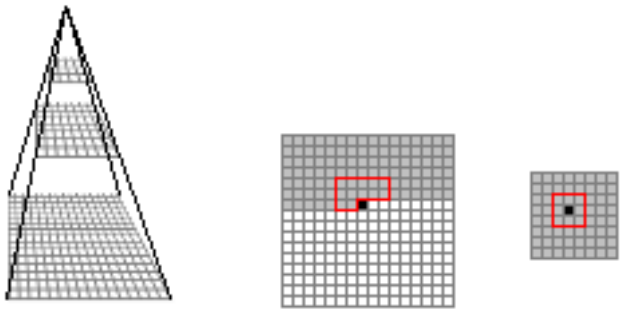
$$\{I_0, ..., I_N\}, \quad I_0 = I, I_{k+1} = G[I_k]$$

Le masque de convolution w doit posséder les propriétés de normalisation, de symétrie, d'unimodalité, d'égale contribution au niveau supérieur et de séparabilité [1]. Parmi les masques impairs, le masque $w = (c\ b\ a\ b\ c)$ avec

$$a = 2/5, \quad b = 1/4 \quad et \quad c = 1/4 - a/2 = 1/20$$

définit ce que l'on appelle une pyramide gaussienne.

Soient G_I et G_O les pyramides gaussiennes [4] de l'image initiale et de l'image de sortie, cette dernière étant initialisée avec du bruit blanc. Au niveau de résolution le plus bas, la méthode décrite dans la section précédente est utilisée. Puis, l'algorithme transforme G_O de la résolution la plus basse vers la plus haute tel que chaque niveau supérieur de résolution soit construit sur base du niveau inférieur déjà synthétisé. À un niveau de résolution R, le voisinage d'un pixel contient désormais deux voisinages mais à des niveaux de résolution différents : celui du niveau R et celui du niveau $R-1$ (voir figure 20). Étant donné qu'au niveau de résolution $R-1$, tous les pixels ont déjà été traités, la forme du voisinage peut-être carrée.



(a) Pyramide gaussienne (b) Niveau R (c) Niveau $R-1$

FIG. 20 – Voisinage à multirésolution 5x5, 3x3.

Désormais des voisinages de faible taille capturent aussi bien l'aspect général de la texture, grâce aux niveaux de résolution les plus bas, que les détails de celle-ci, grâce aux niveaux de résolution les plus hauts. Le temps de traitement est donc considérablement réduit pour des résultats équivalents. Cependant, la recherche par force brute du meilleur voisinage reste le point noir de cette technique.

[4] La pyramide ne doit pas être nécessairement gaussienne. Cependant ce choix a un avantage : il est inutile de reconstruire l'image à la fin de l'algorithme étant donné qu'elle est fournie par le dernier niveau de résolution.

2.4.3 Quantification vectorielle

Un quantificateur vectoriel code les vecteurs à k dimensions de l'espace $\mathbb{R}^k$ par un ensemble fini de vecteur $Y = \{y_i : i = 1, 2, ..., N\}$. Chaque vecteur y_i est appelé mot de code et l'ensemble de tous les mots de code est appelé le dictionnaire. A chaque mot de code y_i est associé la plus proche région, appelée région de Voronoï, définie par :

$$V_i = x \in R^k | \|x - y_i\| < \|x - y_j\|, \forall i \neq j \tag{2.3}$$

L'ensemble des régions de Voronoï divise alors l'espace $\mathbb{R}^k$ tel que :

$$\bigcup_{i=1}^{N} V_i = R^k \qquad \bigcap_{i=1}^{N} V_i = \emptyset \tag{2.4}$$

La figure 21 illustre ces concepts pour des vecteurs à deux dimensions. A chaque mot de code est donc associé un groupe de vecteurs et chaque mot de code possède sa propre région de Voronoï. Etant donné un vecteur d'entrée, le mot de code choisi pour le représenter est celui qui se trouve dans la même région de Voronoï c'est-à-dire que le mot de code représentatif est déterminé pour être le plus proche, en terme de distance Euclidienne, du vecteur d'entrée.

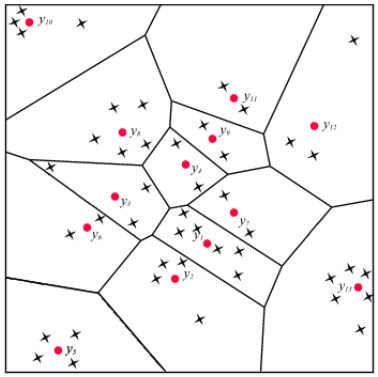


FIG. 21 – Mot de codes à deux dimensions. Les vecteurs d'entrée sont marqués d'une croix, et les mots de code associés d'un cercle rouge. Les régions de Voronoï sont délimitées par des lignes imaginaires.

La quantification vectorielle peut accélérer la recherche du meilleur voisinage dans le cas qui nous occupe. En considérant les voisinages comme des vecteurs à k dimensions, la première étape est donc de constituer le dictionnaire à partir des voisinages inclus dans I et de les arranger sous la forme d'un arbre équilibré [9] pour permettre un parcours rapide. Une fois le dictionnaire calculé, l'opération de recherche du meilleur voisinage pour un pixel donné se réduit alors à la recherche du mot de code le plus proche en parcourant l'arbre. La feuille atteinte donne alors une approximation de la valeur qui aurait été trouvée par une recherche exhaustive.

3 Outils de dessin

Dans cette section, nous allons nous intéresser à la modélisation des différents composants du plumier. Selon les spécifications de Monsieur Leclercq, celui-ci comporte les crayons gris HB et B, le surligneur et le feutre noir. Les crayons étant les outils les plus utilisés par les architectes lors d'une esquisse, c'est par leur modélisation que cette partie du projet débute. Comme nous avons pu le constater dans la section 1, chaque outil de dessin nécessite une étude à part entière de par les différentes propriétés physiques qui le définissent : forme, composition, adhérence au support, ... Par manque de temps, seuls les crayons gris et une gomme ont été implémentés. Cependant, une approche des autres composants du plumier est présentée à la section 3.3 et pourra servir de base lors d'un projet futur.

3.1 Crayon graphite

La mine d'un « crayon gris » est composée de graphite et d'une argile fine, le kaolin. Des liants agglomèrent le mélange et confèrent à la mine sa solidité. Les proportions de graphite et d'argile varient selon le degré de dureté voulu. Plus la mine contient de graphite, plus le crayon est gras. Habituellement, dix-neuf degrés sont utilisés pour caractériser cette dureté. Le tableau ci-contre reprend les proportions de graphite, de kaolin et de liants pour chaque type de crayon.

Notons également que le diamètre de la mine varie de $2\,mm$ pour un crayon 9H à $4\,mm$ pour le 8B et que sa forme dépend de la manière dont le crayon est taillé (figure 23).

Numéro	Graphite	Kaolin	Liants
9H	0.41	0.53	0.06
8H	0.44	0.50	0.06
7H	0.47	0.47	0.06
6H	0.50	0.45	0.06
5H	0.52	0.42	0.06
4H	0.55	0.39	0.06
3H	0.58	0.36	0.06
2H	0.60	0.34	0.06
H	0.63	0.31	0.06
F	0.66	0.28	0.06
HB	0.68	0.26	0.06
B	0.71	0.23	0.06
2B	0.74	0.20	0.06
3B	0.76	0.18	0.06
4B	0.79	0.15	0.06
5B	0.82	0.12	0.06
6B	0.84	0.10	0.06
7B	0.87	0.07	0.06
8B	0.90	0.04	0.06

FIG. 22 – Proportions des différents composants selon le degré de dureté [19].

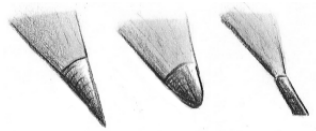


FIG. 23 – Formes de pointes

	Stylet
(X, Y)	Position sur la tablette
P	Pression exercée par l'utilisateur ($\in [0, 1024]$)
α	Angle d'inclinaison selon l'axe des y ($\in [-60\,°, 60\,°]$)
β	Angle d'inclinaison selon l'axe des z ($\in [-180\,°, 180\,°]$)
	Crayon
C	Modèle 3D / 2D du crayon
D_c	Degré de dureté
s_i	Sommet du polygone définissant la pointe
c_{s_i}	Coefficient de distribution de la pression en s_i
$R_{\alpha,\beta}$	Matrice de rotation 3D selon α et β
P'	Pression normalisée exercée sur le stylet
P_{s_i}	Pression exercée sur le sommet s_i
	Feuille
g	Grammage normalisé
h_k	Hauteurs définissant un grain ($k = 1..4$)
V_g	Volume du grain
	Interaction
Q_{max}	Quantité maximale de mine pouvant être déposée dans le volume V_g
P_m	Pression moyenne exercée sur un grain
D	Profondeur de pénétration de la mine dans la volume V_g
Q_{reelle}	Quantité de mine réellement déposée sur un grain
$e(D_c), e'(D_c)$	Facteurs d'échelle
Q_k	Quantité de mine déposée sur la hauteur h_k
G_c, K_c, L_c	Proportions de graphite, de kaolin et de liants
G_k, K_k, L_k	Quantité de graphite, de kaolin et de liants déposée sur la hauteur h_k
I_k	Intensité de gris de la hauteur h_k
Q_{gommee}	Quantité de mine gommée
τ	Taux d'absorption de mine par la gomme

TAB. 1 – Liste des variables utilisées

3.1.1 Modèle 3D

Lors d'un trait de crayon, seule l'extrémité de la mine entre en contact avec la surface du papier. C'est donc assez logiquement que le crayon virtuel sera modélisé par un polyèdre, représentant la forme 3D de la pointe. Un crayon C peut donc se définir comme

$$C = \{D_c,\ s_i = \begin{pmatrix} x_i & y_i & z_i \end{pmatrix} i = 1\,..\,n\} \tag{3.1}$$

où s_i sont les sommets du polyèdre, au nombre de n et D_c est le degré de dureté du crayon. Le but de la modélisation de la mine d'un crayon est d'obtenir une cartographie (*map*) de la pression exercée en chaque point de la pointe du crayon C. Le résultat sera ensuite appliqué en chaque point (X, Y) renvoyé par le stylet.

Un des avantages du dessin virtuel est qu'il n'est plus nécessaire de tailler ses crayons ! Cependant, garder une même pointe tout au long du dessin est une trop grosse approximation de la réalité. Dès lors, pour chaque position (X, Y) renvoyée par le stylet, une nouvelle pointe est générée en cinq étapes :

Étape 1

Génération pseudo-aléatoire d'un polygone convexe à n côtés inscrit dans un cercle dont le rayon, exprimé en pixels, est fonction du degré de dureté du crayon[5], D_c.

$$\begin{array}{l}
\underline{begin} \\
\quad n := 0 \quad ; rayon := \mathsf{f}(D_c) \quad ; angleCentre := \mathsf{random}(45, 60) \\
\quad ; \underline{do} \quad (angleCentre < 360) \rightarrow \\
\qquad n := n + 1 \\
\qquad ; x_n := \mathsf{cos}(angleCentre) * rayon \\
\qquad ; y_n := \mathsf{sin}(angleCentre) * rayon \\
\qquad ; angleCentre := angleCentre + \mathsf{random}(45, 60) \\
\quad \underline{od} \\
\underline{end}
\end{array}$$

Étape 2

Détermination de la profondeur de chaque sommet s_i du polygone ($i \in [1..n]$). Cette profondeur est comprise entre 0 et 1 en fonction de la forme de la pointe voulue. Le centre, s_0, est quant à lui situé à une profondeur égale à l'unité.

[5] et aussi bien évidemment de la résolution de travail

Étape 3

Rotation 3D du polyèdre obtenu à l'étape précédente selon les deux angles d'inclinaison α et β.

$$s_i = s_i \cdot R_{\alpha,\beta} \qquad i = 1..n \tag{3.2}$$

avec

$$R_{\alpha,\beta} = \begin{pmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{pmatrix} \begin{pmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.3}$$

Étape 4

Projection dans le plan (x, y) du polyèdre incliné et détermination des faces susceptibles d'entrer en contact avec la surface de la feuille. Cette étape repose sur les trois notions mathématiques suivantes :

Test d'appartenance d'un point p à un triangle t : Soient s_1, s_2 et s_3 les sommets du triangle t dans le sens trigonométrique. p est intérieur au triangle t si et seulement si les triangles $\triangle s_1 s_2 p$, $\triangle s_1 p s_3$ et $\triangle p s_2 s_3$ ont tous une aire signée négative.

Aire signée A du triangle $\triangle s_1 s_2 s_3$:

$$A = \begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix} \tag{3.4}$$

Calcul de la coordonnée z_p d'un point p appartenant au triangle $\triangle s_1 s_2 s_3$ connaissant x_p et y_p :

$$\begin{vmatrix} x_p - x_1 & y_p - y_1 & z_p - z_1 \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{vmatrix} = 0 \tag{3.5}$$

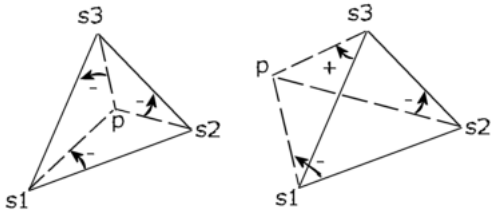


FIG. 24 – Test d'appartenance d'un point p à un triangle $\triangle s_1 s_2 s_3$

Soient alors F l'ensemble des triangles formant les faces du polyèdre projeté dans le plan (x, y)

$$F = \{\triangle s_0 s_1 s_2,\ \triangle s_0 s_2 s_3,\ ...,\ \triangle s_0 s_n s_1\} \tag{3.6}$$

et B le plus petit rectangle dans lequel F peut être inscrit (*bounding box*). Pour chaque point $p \in B$, l'ensemble $F_p \subset F$ de tous les triangles contenant p est construit afin de déterminer les faces du polyèdre susceptibles d'entrer en contact avec la surface de la feuille.

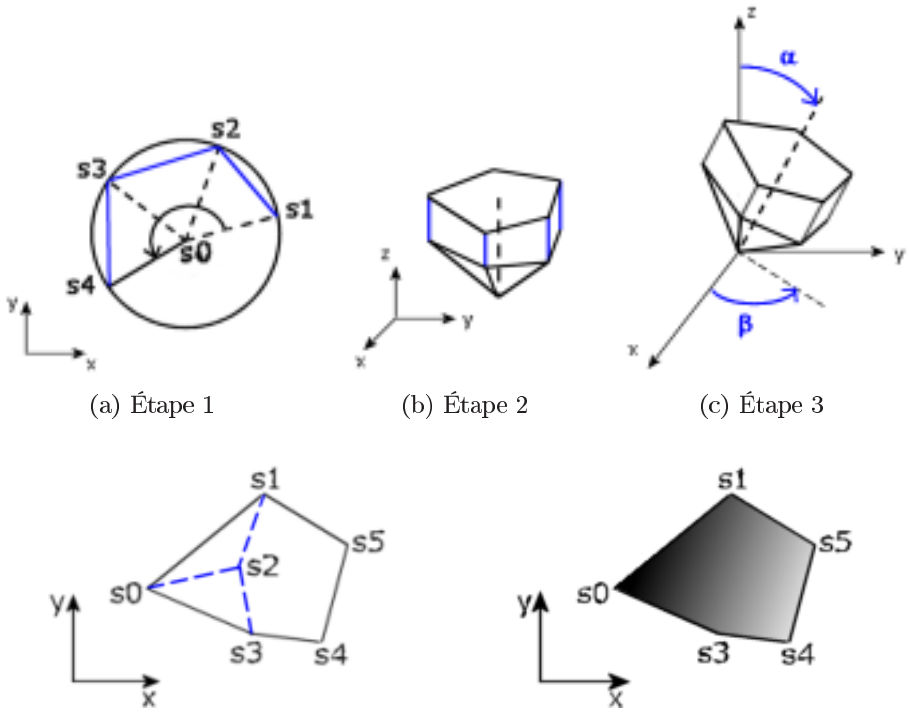
begin
  do ($\forall$ point $p \in B$) $\rightarrow$
    $nb := 0$; $F_p = \emptyset$
    ; *do* ($\forall$ triangle $t \subset F$) $\rightarrow$
      if ($p \in t$) $\rightarrow$
        $nb := nb + 1$
        ; $F_p := F_p \cup \{t\}$
      $\square$ ($p \notin t$) $\rightarrow$ SKIP
      fi
      ; *if* ($nb \leq 1$) $\rightarrow$ SKIP
      $\square$ ($nb > 1$) $\rightarrow$
        t_{min} := "Triangle $\in F_p$ pour lequel z_p est minimum"
        ; $F := F \setminus \{F_p \setminus \{t_{min}\}\}$
      fi
    od
  od
end

Étape 5

Distribution de la pression au travers de la surface projetée de la mine. Pour les sommets des triangles de F, le calcul s'effectue selon

$$P_{s_i} = P' \cdot z_{s_i} \qquad i = 0\,..\,n \tag{3.7}$$
$$P' = \frac{P}{1024} \in [0, 1]$$

et pour tout point intérieur à un des éléments de F, selon l'équation (3.5) où le plan est désormais défini par (x, y, *pression*).



(a) Étape 1 (b) Étape 2 (c) Étape 3

(d) Étape 4 (e) Étape 5

FIG. 25 – Modélisation 3D

3.1.2 Modèle 2D

Après avoir testé l'implémentation de ce modèle 3D, il s'est avéré que le volume de calculs nécessaire à la détermination des faces de la pointe susceptibles d'être en contact avec la surface de la feuille était trop important pour conserver l'aspect temps réel du projet. Étant donné que cette étape ramène à deux dimensions le polyèdre définissant le crayon, modéliser ce dernier directement en 2D constitue une approximation acceptable. Un crayon C est désormais défini par

$$C = \{D_c,\ s_i = \begin{pmatrix} x_i & y_i \end{pmatrix}\ i = 1\,..\,n\} \tag{3.8}$$

Étape 1'

Identique à l'étape 1 du modèle 3D.

Étape 2'

Détermination des coefficients c_{s_i} de distribution de la pression exercée en chaque sommet s_i du polygone ($i = 1\,..\,n$). Ces coefficients sont compris entre 0 et 1 en fonction de la forme de la pointe voulue. Le centre possède quant à lui un coefficient égal à l'unité.

Étape 3'

Soit s_F le sommet du polygone pour lequel

$$y_F = \{ \begin{array}{ll} \mathsf{max}\,(y_i) & si \quad \alpha > 0 \\ \mathsf{min}\,(y_i) & si \quad \alpha < 0 \end{array} \qquad i = 1\,..\,n \tag{3.9}$$

Étirement du polygone proportionnellement à l'angle d'inclinaison α (voir figure 26)

$$y_i = (y_i - y_F) \cdot f(\alpha) + y_F \qquad i = 0\,..\,n \tag{3.10}$$

Et rotation 2D selon l'angle β

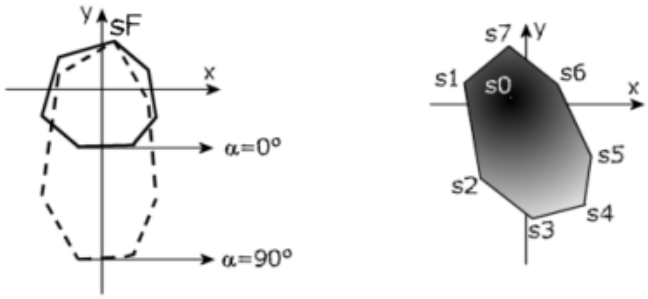
$$s_i = s_i \cdot \begin{pmatrix} \mathsf{cos}\,\beta & -\mathsf{sin}\,\beta \\ \mathsf{sin}\,\beta & \mathsf{cos}\,\beta \end{pmatrix} \qquad i = 0\,..\,n \tag{3.11}$$

Étape 4'

Distribution de la pression au travers de la surface projetée de la mine. Pour les sommets du polygone, le calcul s'effectue selon

$$P_{s_i} = P' \cdot c_{s_i} \qquad i = [0..n] \tag{3.12}$$

où $P' \in [0, 1]$ est la pression P normalisée et pour tout point intérieur au polygone, selon l'équation (3.5) où le plan est désormais défini par (x, y, $pression$).



(a) Étape 3' (b) Étape 4'

FIG. 26 – Modélisation 2D

Remarques

Du point de vue de l'implémentation, une accélération des calculs a été apportée à deux niveaux. Premièrement, étant donné que la valeur des angles d'inclinaison du stylet α et β sont des valeurs entières, les valeurs cos et sin des angles sont extraites d'un tableau de constantes et non calculées à l'aide des fonctions cos() et sin(). En effet, même s'il s'agit là d'un détail, le temps de génération est divisé par un facteur 40. De plus, étant donné qu'il est impossible du point de vue matériel de déceler une rotation du stylet sur lui-même alors que les architectes tracent leurs traits de la sorte afin d'user la mine uniformément, une nouvelle pointe (c'est-à-dire les étapes 1' et 2') est générée pour chaque trait et non pour chaque position (X, Y) du stylet.

3.2 La gomme

Un autre outil de dessin présent dans un plumier et essentiel lors d'une esquisse est bien sûr la gomme. Même si elle n'est pas, dans un premier temps, un des objectifs de ce projet, sa modélisation peut s'apparenter fortement à celle d'un crayon. Pour être plus précis, il faut alors parler de crayon-gomme et non de gomme traditionnelle ou de gomme mastic. Le modèle adopté est un modèle 2D, identique en tout point à celui d'un crayon à l'exception du rayon du polygone généré à l'étape 1'. En effet, ce dernier n'a plus de raison d'être fonction du degré de dureté du crayon et est par conséquent fixé à $3mm$.

(a) Crayon-gomme (b) Traditionnelle (c) Gomme mastic

FIG. 27 – Différents types de gomme

3.3 Extensions

Dans les crayons de couleur, le graphite est remplacé par un pigment minéral ou organique donnant au crayon sa couleur. Dès lors, le modèle présenté ici peut très bien être adapté en substituant la proportion de graphite par celle des composantes RGB du pigment. Bien que fort similaire, l'interaction entre le crayon de couleur et la feuille de papier, devra certainement être quelque peu modifiée. En effet, le rendu de deux traits superposés l'un jaune et l'autre bleu par exemple n'est pas le même suivant l'ordre dans lequel les traits ont été réalisés. Un modèle en couches pourrait être envisagé, la dernière ayant un impact plus important sur la couleur rendue.

Le surligneur quant à lui nécessite une étude à part entière vu le peu de ressemblance avec les propriétés physiques des crayons (la forme géométrique, la non-rigidité de la pointe, ...). De plus, certains effets doivent être pris en compte, tels que la transparence, le mélange des couleurs et le temps de séchage entre chaque trait (figure 28). Des rapprochements pourraient être faits avec l'implémentation du rendu de l'aquarelle dans [4].

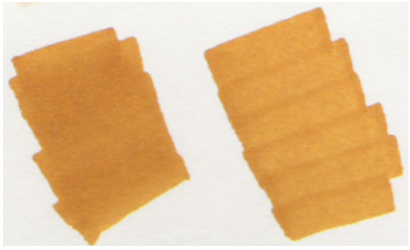


FIG. 28 – Effets obtenus avec un surligneur

4 Interaction avec la feuille virtuelle

4.1 Interaction papier - crayon

Le plus petit élément caractérisant une feuille de papier est son grain. Dans ce projet, la feuille de papier est modélisée par un champ de hauteurs et un grain est donc défini par quatre hauteurs de la feuille (figure 29) : h_1 est situé en (x, y), h_2 en $(x, y+1)$, h_3 en $(x+1, y+1)$ et h_4 en $(x+1, y)$.

Le volume du grain V_g est défini comme étant le volume situé au-dessus du grain

$$V_g = h_{max} - \frac{1}{4}\sum_{k=1}^{4} h_k \tag{4.1}$$

où h_{max} est la plus grande des hauteurs formant le grain considéré. Remarquons que vu la définition des coordonnées des hauteurs du grain, la surface de la base de ce dernier est de 1 pixel $\times$ 1 pixel et par conséquent, V_g représente bel et bien un volume.

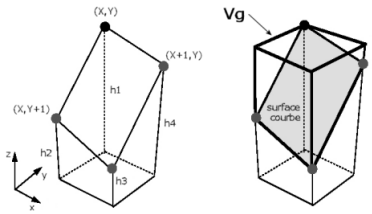


FIG. 29 – Définitions du grain (à gauche) et du volume du grain (à droite)

Après avoir généré une feuille de papier selon l'une des méthodes décrites dans la section 2, l'interaction entre le champ de hauteurs ainsi créé et le crayon est donnée par l'algorithme suivant :

Pour chaque nouveau trait
Générer une nouvelle pointe (Étapes 1' et 2' section 3.1.2)
Pour chaque nouvelle position (X, Y) *du stylet*

1. Inclinaison de la pointe selon α et β (Étape 3' section 3.1.2)
2. Distribution de la pression P (Étape 4' section 3.1.2)
   Pour chaque grain de la feuille interagissant avec la pointe
   (a) Calcul de la quantité maximale de mine (section 4.1.1)
   (b) Calcul de la quantité réelle de mine (section 4.1.2)
   (c) Procéder à l'endommagement de la feuille (section 4.1.3)
3. Calcul de l'intensité du gris (section 4.1.4)

4.1.1 Quantité maximale de mine

La première étape consiste donc à calculer la quantité maximale de mine Q_{max} – c'est-à-dire la quantité maximale de graphite, de kaolin et de liants – pouvant être déposée dans le volume du grain V_g. Deux situations peuvent se présenter :

1. Si toutes les hauteurs h_k du grain sont égales alors Q_{max} est égal à $500pm$ (particules de mine) [6], soit la quantité maximale de mine nécessaire pour remplir la surface plane du grain.

2. Si au moins une des hauteurs h_k du grain est différente des autres alors Q_{max} est donnée par
$$Q_{max} = 1000 + 2000 \cdot V_g \tag{4.2}$$
Par conséquent, dans ce cas, $Q_{max} \in [1000, 3000]\, pm$. En d'autres termes, la quantité de mine déposée sur une surface courbe est plus importante que celle déposée sur une surface plane.

4.1.2 Quantité réelle de mine

La quantité de mine réellement déposée sur chaque grain, Q_{reelle}, est calculée selon les cinq étapes suivantes :

Étape 1. Profondeur de pénétration de la pointe

Plus la pression exercée sur le stylet est élevée, plus la quantité de mine déposée sur le grain est importante. Dès lors, la profondeur D de pénétration de la mine dans le volume du grain (figure 30) est fonction de la pression exercée sur le grain. Cette profondeur de pénétration peut s'exprimer par

$$D = h_{max} - (h_{max} \cdot P_m) \tag{4.3}$$

où P_m est la pression moyenne exercée sur le grain, soit

$$P_m = \frac{1}{4} \sum_{k=1}^{4} P_{h_k} \tag{4.4}$$

Si $D < h_{min}$ alors $D = h_{min}$ où h_{min} est la plus petite des hauteurs formant le grain.

[6] Les valeurs en pm sont basées sur les observations de M. Sousa et J. Buchanan dans [26].

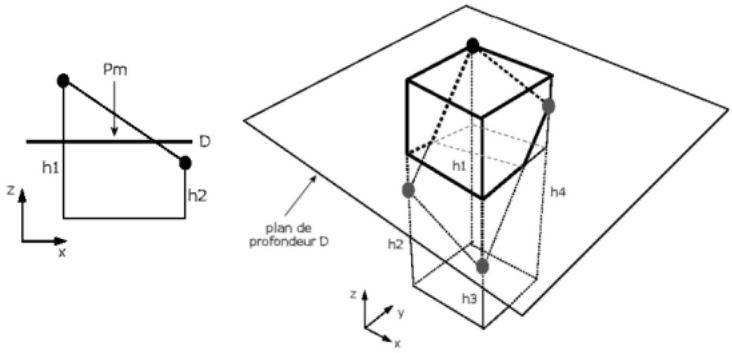


FIG. 30 – Profondeur de pénétration

Étape 2. Quantité réelle de mine

1. Si toutes les hauteurs h_k du grain sont supérieures ou égales à D, c'est-à-dire si la pression appliquée est suffisante pour remplir totalement le volume du grain V_g, alors la quantité réelle de mine est égale à la quantité maximale de mine pouvant être déposée sur le grain
$$Q_{reelle} = Q_{max} \tag{4.5}$$
2. Si par contre au moins une hauteur h_k du grain est inférieure à D, la quantité de mine réellement déposée sera proportionnelle à la profondeur de pénétration D, soit
$$Q_{reelle} = Q_{max} \cdot h_m \tag{4.6}$$
avec
$$h_m = \frac{h_{max} - D}{h_{max} - h_{min}} \tag{4.7}$$

Étape 3. Ajustement selon le degré de dureté du crayon

La quantité de mine réellement déposée sur le grain varie également avec le degré de dureté du crayon utilisé. En effet, toutes autres choses restant égales, plus le crayon est gras, plus la quantité de mine déposée est importante. Dès lors, Q_{reelle} doit être ajustée en fonction du degré dureté D_c

$$Q_{reelle} = Q_{reelle} \cdot e(D_c) \tag{4.8}$$

où $e(D_c) \in [0, 1]$ est un facteur d'échelle : plus le crayon est sec, plus $e(D_c)$ est proche de 0.

Étape 4. Distribution de la quantité de mine sur les hauteurs du grain

La quantité de mine déposée est calculée pour le grain et doit maintenant être répartie proportionnellement sur les différentes hauteurs du grain (figure 31). En effet, plus une des hauteurs du grain h_k est élevée plus elle a tendance à « accrocher » la mine et par conséquent plus la quantité de mine déposée sur cette hauteur, Q_k, est importante.

$$Q_k = Q_{reelle} \cdot hm_k \qquad k = 1..4 \tag{4.9}$$

avec

$$hm_k = \frac{h_k}{\sum_{i=1}^{4} h_i} \tag{4.10}$$

Si $\sum_{i=1}^{4} h_i = 0$ alors $hm_k = 0.25$ pour $k = 1..4$

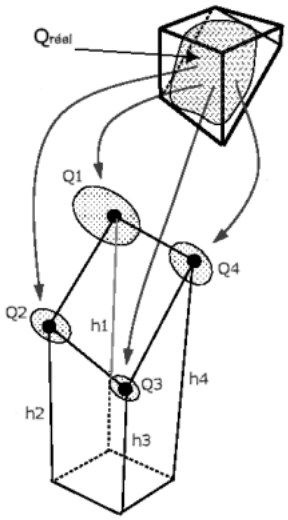


FIG. 31 – Distribution de Q_{reelle} sur les hauteurs h_k du grain

Étape 5. Proportions de graphite, de kaolin et de liants

Soient G_c, K_c et L_c les proportions respectivement de graphite, de kaolin et de liants du crayon utilisé. La quantité totale[7] des composants de la mine déposée après chaque passage du crayon sur la hauteur h_k est alors donnée par

$$G_k = G_k + G_c \cdot Q_k \tag{4.11}$$

$$K_k = K_k + K_c \cdot Q_k \tag{4.12}$$

$$L_k = L_k + L_c \cdot Q_k \tag{4.13}$$

[7] G_k, K_k et L_k, sont donc accumulatifs

4.1.3 Dommages causés à la feuille

Les dommages causés à la feuille par le crayon s'apparentent à un problème de déformation d'un solide sous une contrainte mécanique, et Monsieur le Professeur Serge Cescotto, du département de mécanique des matériaux, était tout désigné pour me guider dans ma réflexion. Trois pistes ressortent de nos entrevues. La première consiste en une étude mathématique de la déformation du papier sous la contrainte de la pression appliquée au crayon. Malheureusement, une telle étude nécessite de connaître le graphe contrainte-déformation pour chaque type de papier, données inexistantes chez les principaux fabricants de papier à dessin. Seules les entreprises de fabrication de papier d'emballage semblent concernées par ce type de graphes, qu'elles exploitent pour optimiser la vitesse de passage du papier dans leurs chaînes de production. Quand bien même ces graphes seraient en notre possession, les hypothèses qui doivent être faites dans le calcul mathématique de la déformation de la feuille sont trop restrictives. En effet, elles supposent la feuille lisse et la pointe du crayon verticale et de révolution.

La seconde piste suivie est la mesure de la déformation d'une feuille au microscope optique. Pour ce faire, une feuille vierge est photographiée, puis quelques traits y sont dessinés à l'aide d'un stylet électronique dont la pointe a été remplacée par une vraie mine de graphite. Ainsi, une correspondance peut être établie entre un trait réel et la pression à laquelle il a été réalisé. Ensuite, ces traits sont gommés et la feuille est à nouveau photographiée. Une comparaison peut alors être effectuée avec la première image. Après une série de tests, il apparaît que les déformations ne sont visibles que pour des pressions de l'ordre de 1000 (figure 32) – pour un maximum de 1024 pouvant être renvoyé par le stylet – ce qui n'est évidemment pas représentatif.

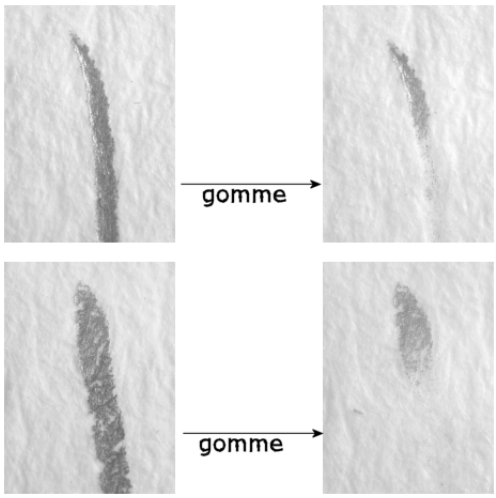


FIG. 32 – Étude au microscope optique sur papier Canson C à grains - en haut : pression moyenne de 1000, en bas : pression moyenne de 426

Au vu des résultats obtenus jusqu'à présent, c'est une approche plus intuitive qui a été adoptée. En effet, quels sont les paramètres à prendre en compte pour simuler une déformation ? En supposant que la feuille soit placée sur un support rigide, parfaitement lisse et indéformable, il s'agit essentiellement du type de crayon (un crayon sec abîme plus la feuille qu'un crayon gras), de l'épaisseur de la feuille (son grammage plus exactement) et de la pression exercée. La hauteur du grain après déformation peut donc s'exprimer par

$$h_k = h_k - P_m \cdot e'(D_c) \cdot g \tag{4.14}$$

où $g \in [0, 1]$ est le grammage normalisé de la feuille et $e'(D_c) \in [0, 1]$ est un facteur d'échelle – plus le crayon est sec, plus il endommage la feuille et plus $a(D_c)$ est proche de 1.

4.1.4 Intensité de gris

Finalement, les particules de kaolin et de liants étant transparentes, l'intensité de gris I_k dépend de la proportion de particules de graphite présente en h_k et s'exprime par

$$I_k = 1 - \frac{G_k}{Q_{max}} \tag{4.15}$$

4.2 Interaction papier - gomme

Sous l'hypothèse que le passage de la gomme sur la feuille n'endommage pas celle-ci, l'algorithme d'interaction entre le papier et la gomme est semblable à celui de l'interaction entre la feuille et le crayon (section 4.1), excepté les points (b) et (c) qui sont remplacés par le calcul de la quantité de mine gommée, Q_{gommee}. Cette dernière est non seulement proportionnelle à la quantité de graphite présente dans le volume d'un grain donné mais également à la pression exercée sur la gomme. De plus, il faut tenir compte du taux d'absorption de mine par la gomme. En effet, ce dernier varie selon le procédé de fabrication et la composition de la gomme. Dès lors, la quantité de mine gommée s'exprime par

$$Q_{gommee} = G_k \cdot P_m \cdot \tau \tag{4.16}$$

où $\tau \in [0, 1]$ est le taux d'absorption de mine. Selon les observations de M. Sousa et et J. Buchanan dans [26], τ est de l'ordre de 0.7 pour une gomme de type traditionnel. La quantité de graphite restant sur la hauteur h_k est alors donnée par

$$G_k = G_k - Q_{gommee} \tag{4.17}$$

La gomme implémentée efface véritablement le graphite déposé, au contraire des tortillons, estompes et autres gommes mastic qui étalent et/ou estompent les traits de crayon. Lors du passage de la gomme sur le papier, il n'y a donc transfert de particules de mine que du papier vers la gomme.

5 Prototype et résultats

Le prototype élaboré écrit en ANSI C et développé sous Dev-Cpp utilise les librairies OpenGL pour le graphisme, OpenGl Utility Toolbox pour la gestion des événements, OpenGL User Inetrface pour la gestion de l'interface utilisateur et enfin JPEG pour la sauvegarde des esquisses. Le choix de ces librairies a été guidé par les contraintes qu'imposait le système *EsQUIsE*. En effet, chacune d'entre elles est compatible avec les systèmes Mac, Win32 et Unix.

Chaque section, voire sous-section, de cet ouvrage constitue un module de développement à part entière, testé et optimisé de manière indépendante. Le prototype final résulte d'un assemblage de ces modules au sein d'une interface graphique permettant à l'utilisateur d'interagir avec le prototype les plus intuitivement possible. L'interface utilisateur est en conséquence très simple et uniquement constituée d'un espace de dessin, la feuille virtuelle, et d'un plumier permettant de sélectionner un outil de dessin (figure 33).

Un menu pop-up est également accessible et permet à l'utilisateur de changer de mode de dessin, de créer une nouvelle feuille virtuelle ou enfin de sauvegarder le dessin en cours au format JPEG. Complémentairement à l'utilisation de la tablette graphique, deux autres modes de dessin sont possibles. Le premier permet de dessiner à l'aide d'une souris, en spécifiant manuellement les angles d'inclinaison et la pression exercée sur le stylet. Le deuxième mode permet de tracer un croquis automatiquement à partir d'un fichier généré par le système *EsQUIsE*. La création d'une nouvelle feuille implique pour l'utilisateur la spécification d'une taille et d'une technique de génération. Cette dernière peut en outre nécessiter un paramétrage plus détaillé selon la méthode utilisée (figure 34).

La figure 35 propose une comparaison du rendu des traits feuille à feuille ainsi que des principaux degrés de dureté des crayons. Le piètre rendu des traits dessinés sur une feuille générée pseudo-aléatoirement justifie la mise en œuvre de techniques plus élaborées et réaffirme l'importance de la qualité de modélisation d'une feuille virtuelle. Le faible réalisme des traits obtenus avec Perlin n'est pas surprenant. En effet, cette méthode ne fournissait déjà pas une texture visuellement semblable à celle obtenue pour de vraies feuilles de dessin. La technique du Quilting tient quant à elle toutes ses promesses et en se reportant à la figure 8, les concordances entre les différentes textures de feuille et le rendu des traits apparaissent nettement. Citons par exemple la feuille Fabriano WaterColor dont la rugosité très prononcée « accroche » la mine au passage du crayon permettant à une quantité importante de graphite de se déposer.

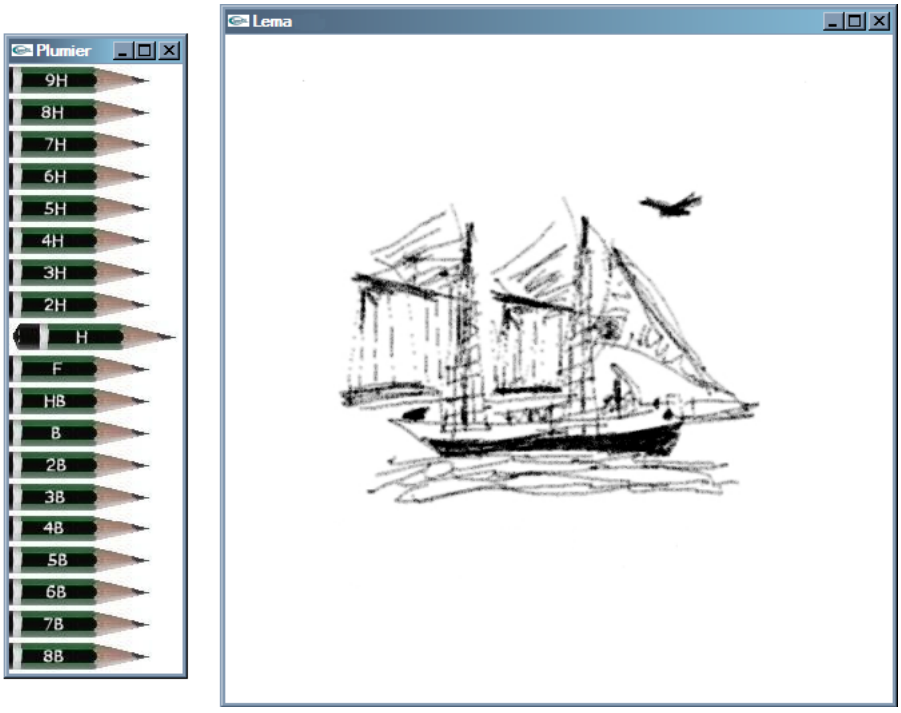


FIG. 33 – Interface utilisateur

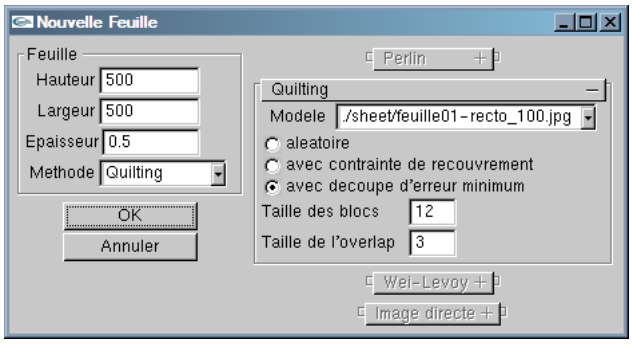


FIG. 34 – Menu « Nouvelle Feuille »

(a) Aléatoire

(b) Perlin

(c) Quilting - Canson C à grains

(d) Quilting - Canson Bristol

(e) Quilting - Canson Mi-teinte ocre

(f) Quilting - Fabriano WaterColor

(g) Quilting - ClaireFontaine (huile)

FIG. 35 – Résultats obtenus sur différentes feuilles virtuelles pour les crayons 6H, 4H, 2H, H, F, HB, B, 2B, 4B et 6B (de gauche à droite)

6 Perspectives

Ce projet novateur n'en est qu'à ses débuts et bien des perspectives lui sont ouvertes. Si certaines d'entre-elles ont déjà été évoquées dans ce document, d'autres méritent d'être évoquées et une synthèse s'impose donc.

L'objectif premier de ce travail ne consistant pas uniquement en la modélisation d'une feuille de papier, certaines techniques d'acquisition de données n'ont pu être explorées. Il serait ainsi intéressant d'obtenir les résultats de l'interférométrie de speckel afin de caractériser plus correctement les variations du champ de hauteurs que constitue le modèle d'une feuille de dessin virtuelle. En ce qui concerne la génération d'une feuille virtuelle, l'implémentation de l'algorithme de Wei-Lewoy [28] en utilisant la quantification vectorielle permettrait un gain de temps considérable.

Les perspectives relatives aux outils de dessin sont beaucoup plus nombreuses, si l'on tient compte des différents pinceaux, fusains et autres feutres qui pourraient être modélisés. Une première amélioration du prototype serait l'intégration de la couleur pour les crayons. Les nouveaux outils ainsi modélisés devront tenir compte de la manière dont les couleurs se mélangent, et un modèle en couches sera certainement nécessaire. Toute aussi concernée par les couleurs et leur mélange, voire leur transparence, l'implémentation d'un surligneur devra être faite en parallèle.

Même si les gommes mastic sont plus couramment utilisées par les dessinateurs que par les architectes, leur implémentation pourrait être envisagée, surtout si les produits sur lesquels débouche le prototype ne s'appliquent pas uniquement à la conception architecturale. Dans le même ordre d'idée, pourquoi ne pas imaginer une interaction directe entre le doigt de l'utilisateur et le dessin, s'il désire en estomper les contours par exemple. A cette fin, un autre travail de fin d'études également réalisé cette année en collaboration avec *LEMA* pourrait se révéler utile. En effet, l'objectif de celui-ci est la détection du contact d'un doigt sur une surface de dessin virtuel.

A plus court terme, l'intégration du prototype dans le logiciel *EsQUIsE* est envisagée afin d'y tester ses performances temps réel. En effet, si jusqu'à présent le prototype conserve bien ce caractère essentiel du projet, il en sera peut-être autrement lorsqu'il sera placé en parallèle des traitements réalisés par *EsQUIsE*.

Conclusion

Ce document présente un algorithme temps réel de rendu non-photoréaliste pour un environnement de dessin virtuel. Ces spécifications font de ce travail de fin d'études un projet sans précédent. La documentation en la matière est par conséquent réduite et la réflexion s'est donc appuyée sur des articles traitant de sujets soit similaires mais moins contraignants, soit complètement différents dont les concepts ont été détournés de leur utilité première. Ainsi, la génération d'une feuille virtuelle se fait par des techniques de synthèse de texture. Ces dernières nécessitent une image de base à partir de laquelle la nouvelle texture est générée. L'acquisition d'un modèle de référence d'une feuille de dessin constitue donc la première étape concrète de ce travail.

Une fois la feuille virtuelle générée, à peine la moitié du chemin était parcouru. En effet, il fallait encore modéliser les outils de dessin et leur interaction avec cette feuille virtuelle. Étant donné que dans la littérature un article n'est dédié qu'à un seul élément du plumier, la tâche était ardue et les deux outils essentiels lors d'une esquisse ont été privilégiés : les crayons gris et la gomme.

La dernière étape fut l'assemblage de ces différentes parties au sein d'une interface graphique simple et intuitive. Au vu des résultats obtenus, le temps passé à caractériser au plus près de la réalité les différents composants ne fut pas vain.

Bref, un projet absorbant mais passionnant car novateur et différent de tous ceux qu'il m'a été demandé de réaliser tout au long de mes études où les bases théoriques et le cadre de travail étaient fixés.

Références

[1] P. Burt et E. Adelson The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communication 31*, pages 532–540, 1983.

[2] B. Cabral et L. Leedom Imaging Vector Fields Using Line Integral Convolution *Computer Graphics Annual Conference Series*, pages 263–270, 1993.

[3] Centre Spatial de Liège Université de Liège *http ://www.ulg.ac.be/cslulg/home.html*

[4] C. Curtis, S. Anderson, J. Seims, K. Fleischer, et D. Salesin Computer-generated watercolor. *Proceedings of SIGGRAPH 97* pages 421–430, Août 1997.

[5] Department of Materials Science & Engineering Iowa State University *http ://mse.iastate.edu/microscopy/home.html*

[6] Bloodshed Dev-Cpp, An Integrated Development Environment *http ://www.bloodshed.net/*

[7] A. Efros et W. Freeman Image Quilting for Texture Synthesis and Transfer. *Proceedings of SIGGRAPH 01*, pages 341–346, Août 2001.

[8] J. Foley, A. vanDam, S. Feiner et J. Hughes Computer Graphics : Principles and Practice. Addison Wesley Publishing Company, 1993.

[9] A. Gersho et R. Gray Vector Quantization and Signal Compression. *Kluwer Academic Publisher*, 1992.

[10] OpenGL User Toolbox Documentation *http ://pyopengl.sourceforge.net/documentation/manual/reference-GLUT.html*

[11] P. Hilton Imaging Surface Roughness using Correlated Speckle Grain Pairs. *http ://www.is.irl.cri.nz/pubdoc/1997/dicta97-pjh.pdf*

[12] Laboratoire d'études méthodologiques architecturales Université de Liège *http ://www.lema.ulg.ac.be*

[13] L. Liang, C. Liu, Y. Xu, B. Guo et H.-Y. Shum Real-time texture synthesis by patch-based sampling. *Technical Report MSR-TR-2001-40 Microsoft Research* , Mars 2001.

[14] A. Linde et R. Gray, An algorithm for vector quantization design. *IEEE Transactions on Communicatinos COM-28*, pages 84–95, Janvier 1980.

[15] X. Mao, Y. Nagasaka et A. Imamiya Automatic Generation of Pencil Drawing Using LIC.*Proceedings of SIGGRAPH 02*, page 149, 2002.

[16] OpenGL Documentation *http ://www.opengl.org/*

[17] K. Perlin An image synthesizer. *Proceedings of SIGGRAPH 85*, pages 287–296, Juillet 1985.

[18] U. Persson Roughness measurement of machined surfaces by means of the speckle technique in the visible and infrared regions. *Optical Engineering 32(12)*, pages 3327–3332, Décembre 1993.

[19] H. Petroski The Pencil : A History of Design and Circumstance. *Knopf Edition*, Janvier 1990.

[20] Piranesi *Informatix Software International http ://www.informatix.co.uk/piranesi/index.shtml*

[21] E. Praun, A. Finkelstein et H. Hoppe Lapped textures. *Proceedings of SIGGRAPH 00*, pages 465–470, 2000.

[22] E. Praun, H. Hoppe, M. Webb et A. Finkelstein Real-time hatching. *Proceedings of SIGGRAPH 01*, 2001.

[23] P. Rademacher GLUI, A GLUT-Based User Interface Library *http ://www.cs.unc.edu/ rademach/glui/src/release/glui_ manual_ v2_ beta.pdf*

[24] C. Reynolds Stylized Depiction in Computer Graphics Non-Photorealistic, Painterly and 'Toon Rendering, an annotated survey of online resources. *http ://www.red3d.com/cwr/npr*

[25] M. Salisbury, S. Anderson, R. Barzel, et D. Salesin Interactive pen-and-ink illustration. *Proceedings of SIGGRAPH 94* pages 101–108, Juillet 1994.

[26] M. Sousa et J. Buchanan. Computer-generated graphite pencil rendering of 3D polygonal models. *Computer Graphics Forum (Eurographics '99)*, volume 18(3), pages 195–208, 1999

[27] B. Wang An algorithm for Detecting Optimized Patch Size For Image Quilting.

[28] L. Wei et M. Levoy Fast Texture Synthesis using Tree-structured Vector Quantization. *Proceedings of SIGGRAPH 00*, pages 479–488, 2000.

[29] G. Winkenback et D. Salesin Computer-generated pen-and-ink illustration. *Proceedings of SIGGRAPH 94*, pages 91–100, Juillet 1994.

[30] Y. Xu, B. Guo et H.-Y. Shum Chaos mosaic : Fast and memory efficient texture synthesis. *Technical Report MSR-TR-2000-32 Microsoft Research*, Avril 2000.