Création semi-automatique de modèles numériques de terrains - Visualisation et interaction sur terminaux mobiles communicants

Joachim Pouderoux

N° d'ordre : 3396

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ BORDEAUX 1

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par **Joachim Pouderoux**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Création semi-automatique de modèles numériques de terrains
Visualisation et interaction sur terminaux mobiles communicants

---

Soutenue le : 29 juin 2007

Après avis des rapporteurs :

Marc Pierrot-Deseilligny	Ingénieur HDR
Jean-Pierre Jessel	Professeur

Devant la commission d'examen composée de :

Marc Pierrot-Deseilligny	Ingénieur HDR	Rapporteur
Jean-Pierre Jessel	Professeur	Rapporteur
Christophe Schlick	Professeur	Président
Kadi Bouatouch	Professeur	Examinateur
Jean-Christophe Gonzato	Maître de Conférences	Examinateur
Pascal Guitton	Professeur	Examinateur

– 2007 –

Création semi-automatique de modèles numériques de terrains Visualisation et interaction sur terminaux mobiles communicants

Résumé :

Les modèles numériques de terrains (MNT) permettent de représenter efficacement la topographie d'une zone géographique donnée. Ces MNT peuvent être issus d'une télé acquisition à l'aide de radars embarqués ou de levés manuels sur le terrain à l'aide d'un tachéomètre. Les MNT constituent le contexte thématique de cette thèse qui est composée en trois parties.

Dans un premier temps, nous nous intéressons à la création de ces modèles à partir d'une source importante de données topographiques constituée par les cartes topographiques. Nous présentons une chaîne complète de traitements permettant de générer un MNT à partir d'une carte topographique numérisée. Nous détaillons particulièrement de nouvelles méthodes de reconstruction des courbes de niveaux et d'interpolation de ces courbes pour générer un MNT. Les différents travaux effectués dans cette thématique s'intègrent au sein de la plate-forme logicielle AutoDEM que nous avons développée durant cette thèse.

Puis, dans une deuxième partie, nous présentons une nouvelle technique permettant de visualiser des MNT en 3D sur une large gamme de dispositifs allant de stations de travail reliées à de grands écrans jusqu'à des terminaux mobiles (TM) à faibles capacités tels que les PDA ou les téléphones portables. L'intérêt majeur de la technique présentée, qui repose sur un mode connecté client-serveur, réside dans l'adaptation dynamique du modèle 3D aux capacités d'affichage du terminal. Nous nous intéressons également à des techniques de rendu à distance et présentons deux techniques permettant d'offrir d'une part une visualisation interactive temps réel et d'autre part un panorama virtuel à l'utilisateur.

Enfin, dans un troisième temps, nous décrivons des techniques nouvelles permettant à un utilisateur mobile disposant d'un TM de naviguer et d'interagir avec des données géographiques (cartes ou plans 2D et scènes 3D). La première est une technique d'interaction tangible et bi-manuelle reposant sur la détection par analyse du flux vidéo d'une cible décrivant un code couleur. La deuxième est une technique de sélection à deux niveaux adaptée aux TM ne disposant pas de dispositif de pointage continu.

Discipline : Informatique

Mots-Clefs : Système d'information géographique, Carte topographique, Analyse d'image, Modèle numérique de terrain, Visualisation 3D, Terminaux mobiles communicants, Interface homme-machine

LaBRI - INRIA
Université Bordeaux 1
351 cours de la Libération
33405 Talence Cedex (FRANCE).

Semi-Automatic Creation of Digital Terrain Models Visualization and Interaction on Handheld Computers

Abstract :

Digital terrain models (DTM) allow to represent efficiently the topography of a given geographical area. Those DTM can be issued from a remote acquisition using, for example, embedded radars or from a topographic surveying with a tacheometer. DTM are the thematic context of this thesis which is composed of three parts.

In a first part, we take an interest in the creation of such models from an important source of topographical data constituted by the topographic maps. We present a complete workflow of treatments to generate DTM from a scanned topographic map. We focus especially on new methods for contour lines reconstruction and for DTM interpolation from a set of contour lines. All the work led in this area was integrated into a software framework called AutoDEM, specifically developed during this PhD.

Then, in a second part, we present a new technique to visualize DTM in 3 dimensions on a wide range of devices going from a workstation plugged to large displays to a handheld device with low capacities like a PDA or a cell phone. The major interest of the presented technique, which is based on a connected client-server mode, is the dynamic adaptation of the large 3D model to the terminal's potential. We also inspect some remote rendering techniques and present two new ones which allow to make a real time interactive visualization, and to offer a virtual interactive panorama service to the user.

Finally, in a third part, we describe new techniques which allow a mobile user featuring a handheld computer to navigate and interact with geographical data (2D maps or 3D scenes). The first is a tangible and bi-manual interaction technique based on the detection of a specific color-coded target by analyzing the video flow. The second one is a two selection levels technique suited to handhelds which do not provide a continuous pointing device.

Discipline : Computer-Science

Keywords : Geographical information system, Topographic map, Image analysis, Digital terrain model, 3D visualization, Handheld computers, Human-computer interaction

LaBRI,
Université Bordeaux 1,
351 cours de la Libération,
33405 Talence Cedex (FRANCE).

Remerciements

Personne n'a été autant que moi persuadé de la futilité de tout,
personne non plus n'aura pris au tragique un si grand nombre de choses futiles.
Emil Michel Cioran

Je tiens tout d'abord à remercier mes directeurs de thèse, Pascal Guitton et Jean-Christophe Gonzato, pour la confiance qu'ils m'ont faite en acceptant de m'encadrer. Je tiens aussi à leur signifier toute ma gratitude pour m'avoir obtenu un financement inespéré sans lequel la poursuite de ce doctorat aurait peut-être été compromise ou, pour le moins, rendue bien plus difficile. Il me faut donc également remercier la région Aquitaine et les fonds sociaux européens qui ont financé les trois premières années de mes recherches.

Trouver un jury pour évaluer un travail ayant un si large spectre n'était pas chose évidente. Je remercie les membres de ce jury pour l'expertise portée dans des domaines si différents. Merci à Marc-Pierrot Deseilligny et Jean-Pierre Jessel qui m'ont fait l'honneur de rapporter cette thèse. Merci à Kadi Bouatouch de m'avoir fait le plaisir de participer à ce jury et à Christophe Schlick pour l'avoir présidé avec le talent qu'on lui connait.

Les travaux trop rapidement synthétisés dans ce mémoire ne sont pas le fruit d'un travail purement individuel, mais sont, pour la plupart, issus de collaborations. Par ordre d'apparition, je dis toute ma gratitude à ceux qui m'ont accompagné sur un ou plusieurs travaux. Merci infiniment à Salvatore Spinello sans qui cette thèse n'aurait peut-être jamais abouti. Merci Salvatore pour tes encouragements, tes lumineuses idées et ces nombreux samedi ou dimanche après-midi passés à assouvir, autour d'un thé chaud, notre passion commune pour les courbes de niveau ! Merci à Irek Tobor d'avoir partagé son expertise sur la reconstruction par FBR et de m'avoir légué - trop tôt - sa persistante plante verte qui continue d'apaiser mon espace de travail. Merci énormément à Martin Hachet de m'avoir impliqué dans ses travaux relatifs à l'interaction, souvent dans une bonne humeur allant jusqu'aux plus incontrôlables fous rires. Enfin, malgré la tournure tragique de notre relation, merci à Jean-Eudes Marvie pour cette trop courte période de forte complicité et de collaboration finalement fructueuse.

Comment aurais-je vécu ces presque quatre années de doctorat sans mes compères d'infortune ? J'aimerais d'abord citer l'éminent Matthias Robine, avec qui j'ai passé des moments inoubliables, depuis la licence, à refaire le monde politique, le système universitaire ou le livre des inventions. Viennent ensuite le bandit velléitaire François De Vieilleville, l'haltérophile valeureux Romain Pacanowski et l'insatisfait multitâche Mickaël Raynaud.
Je tiens aussi à saluer chaleureusement tous les camarades croisés sur ce chemin de croix, parmi eux : Florian Levet avec qui j'ai partagé un modeste vestibule et les débuts de l'aven-

ture du doctorat au sein de l'équipe Iparla, Jocelyn Fréchot pour notre point commun, Fabrice Décle pour sa bonne humeur, Sebastian Knoëdel pour son joyeux français, Julien Hadim pour son paradoxal système orexinergique, Jean-Charles Quillet pour son gaufrier, Audrey Legeai et Jérôme Baril pour leur discrétion.

Je remercie infiniment mon plus cher ami, mon fréro ad-vitam, mon camarade de dégustation 'pâtière', j'ai nommé le Dr. Damien Thomas Achard. Merci à toi pour toutes ces années d'infinies complicité et amitié... Plût au ciel qu'elles durent à jamais.
Je n'oublie pas non plus mes amis plus lointains : Laurent Delabre l'otaku philosophe, Yannick Montet le motard surgelé et Tam Putoa le casanier geek tahitien. Plus lointain encore je pense à tout mes amis d'enfance puis d'adolescence qui ont partagé avec moi la passion de l'informatique ; la liste serait longue mais je n'oublie pas mes vieux amis Fred, Serge et Emmanuel.

Enfin, aurais-je pu parcourir toutes ces années d'études sans les supports affectif et matériel de mes proches ? J'adresse donc ma plus profonde gratitude à mes parents, grands parents, sœurs et beaux parents, au sens très large. A Pélagie, je dis ma plus délicate affection, pour toutes ces années de bonheur partagé.

Table des matières

Table des figures

Liste des tableaux

Introduction

Contexte et enjeux

Ces dernières années ont connu l'essor de nouvelles technologies de visualisation de l'information géographique. Celles-ci ont profité d'une part des nouvelles structures de communication, dont Internet est bien entendu la figure de proue, et d'autre part de technologies de positionnement par satellite, en particulier grâce au GPS (*Global Positioning System*) américain. De nouvelles applications géographiques grand public embarquées sur des terminaux mobiles (TM) ou sur des stations de travail ont vu le jour : dans un premier temps, des applications ou sites internet ont proposé la visualisation en 2D de cartes routières vectorielles ou des cartes topographiques. Dans un second temps, sont apparues des applications permettant de visualiser des données en 3D issues d'images et modèles de terrains de la terre entière acquis par satellite (NASA WorldWind, KeyHole puis Google Earth). Ces applications offrent des possibilités toujours accrues par le biais d'images de plus en plus précises, la possibilité d'afficher des couches d'information supplémentaires (cartes, routes, noms de lieux, etc.).

Dans le cadre d'applications de réalité virtuelle, un aspect important repose sur la capacité de reproduire la topographie de la région visualisée en 3 dimensions. Cette la description topographique se réalise à l'aide d'un modèle numérique de terrain (MNT) qui contient un ensemble de points de la surface représentée et leur altitude respective. Si les nouvelles technologies d'acquisition de la topographie via des radars embarqués permettent, aujourd'hui, de générer des modèles de terrains relativement fidèles pour toutes les régions du globe, leur coût reste un frein à une utilisation très localisée. Dans des domaines qui s'intéressent à l'histoire géologique, humaine, architecturale, etc., la reproduction de la topographie ancienne d'un territoire joue un rôle important. Depuis plusieurs siècles, la cartographie, en offrant une vue stratégique des territoires, a été un enjeu majeur des civilisations humaines. Les cartes topographiques, caractérisées par leur représentation schématique à l'aide de courbes de niveau, ont longtemps été le seul moyen d'appréhender les caractéristiques morphologiques d'un territoire. Ces cartes constituent donc une ressource d'informations gigantesque sur l'état actuel ou ancien de la Terre, et l'extraction automatisée de ces informations un challenge important.

Cette thèse s'est effectuée au sein du projet Iparla (LaBRI - INRIA) qui a pour thématique de recherche principale la conception de techniques de modélisation, de visualisation et d'interaction de contenus 3D dans un cadre de mobilité. Le terme mobilité repose sur l'utilisation

des matériels informatiques mobiles, que nous appelons terminaux mobiles communicants (TMC) et qui désignent les ordinateurs de poche tels que les assistants personnels (PDA) ou les téléphones portables. Les méthodes développées dans le cadre du projet Iparla cherchent à prendre en compte les différentes contraintes de ces terminaux (telles que leur faible capacité de calcul, la taille réduite de leur écran ou leur interface d'entrée restreinte) mais également leurs atouts (tels que leur mobilité, leur capacité de connexion à des réseaux WiFi, GSM, etc.).

Objectifs

L'objectif initial de cette thèse était de proposer un ensemble de techniques pour créer des MNT, puis pour naviguer dans ces MNT en 3D sur des TMC.

Notre première démarche a consisté à mettre au point une chaîne de traitement semi-automatique pour générer des MNT. Il s'agissait d'établir un état des lieux des techniques existantes et de contribuer à augmenter l'automatisation du processus à l'aide de techniques nouvelles et d'un logiciel adapté.

La visualisation de grands MNT en 3D sur des plates-formes limitées comme des TMC, était une seconde étape particulièrement importante dans le cadre du projet Iparla. Il s'agissait de mettre en place une technique pouvant s'adapter aux capacités hétérogènes des différentes plates-formes utilisées dans le projet.

Enfin, la navigation dans des informations géographiques 2D ou 3D à l'aide des interfaces offertes par défaut sur les TMC étant délicate, nous avons cherché dans un troisième temps à proposer des techniques d'interaction adaptées.

Plan du mémoire

Le présent mémoire se décompose en trois grandes parties.

La partie I aborde le problème de la création semi-automatique de MNT à partir de cartes topographiques et présente, après les définitions et concepts de la géomatique (chapitre 1), les différentes étapes (chapitres 2, 3 et 4) de la chaîne proposée. Les différentes recherches menées dans ce domaine ont été regroupées au sein d'un logiciel spécifique que nous avons développé et qui est présenté dans le chapitre 5.

Dans la partie II, nous nous intéressons au problème de la visualisation des MNT sur des terminaux. Nous présentons dans le chapitre 6, notre technique de rendu adaptatif en 3D sur des plates-formes hétérogènes de vastes modèles de terrains, stockés sur un serveur distant. Dans le chapitre 7 nous décrivons deux techniques de rendu à distance où le TMC ne fait office que de plate-forme d'interaction et d'affichage de l'image 3D calculée sur un serveur dédié.

Enfin, la partie III présente deux techniques d'interaction pour TMC que nous avons mises au point. La première, décrite dans le chapitre 8 présente une technique d'interaction tangible à 3 degrés de liberté. La seconde technique, baptisée *Jump and Refine*, est présentée dans le chapitre 9 et décrit un schéma de pointage à l'aide d'entrées discrètes (joystick ou touches) adapté aux TMC ne disposant pas d'interface de pointage direct (stylet ou écran tactile).

Première partie

Création de modèles numériques de terrains

FIG. 1: *Table de numérisation utilisée pour numériser les données cartographiques.*

DEPUIS une vingtaine d'années, l'extraction d'informations sur des cartes topographiques numérisées a été l'objet de recherches intensives. Pendant longtemps, cette extraction s'est faite manuellement à l'aide de tables de numérisation (voir figure 1). De longues heures de travail étaient alors nécessaires pour numériser les courbes de niveau des cartes. Si cette technique manuelle se pratique toujours, l'outil informatique propose désormais des capacités de traitement suffisantes pour automatiser tout ou partie du traitement.

Le but de nos investigations dans cette thématique était de proposer une chaîne de traitement la plus automatique possible. Après avoir évalué les différentes techniques ayant été proposées, nous nous sommes rapidement rendu compte que cette tâche était bien plus complexe qu'elle n'en avait l'air si le but était de mettre au point une méthodologie efficace quelque que soit la carte analysée.

Nous avons décomposé cette problématique générale en quatre grandes parties, illustrées par la figure 2 :

1. extraction des courbes de niveau de la carte par segmentation ;
2. reconstruction automatique ou semi-automatique des courbes interrompues ;
3. affectation ou cotation semi-automatique des altitudes des courbes ;
4. création d'un modèle numérique de terrain par interpolation des courbes de niveau.

Dans le chapitre 1, nous décrivons les notions de base de la géomatique et du domaine des systèmes d'information géographique qui seront utilisés dans la suite du document.

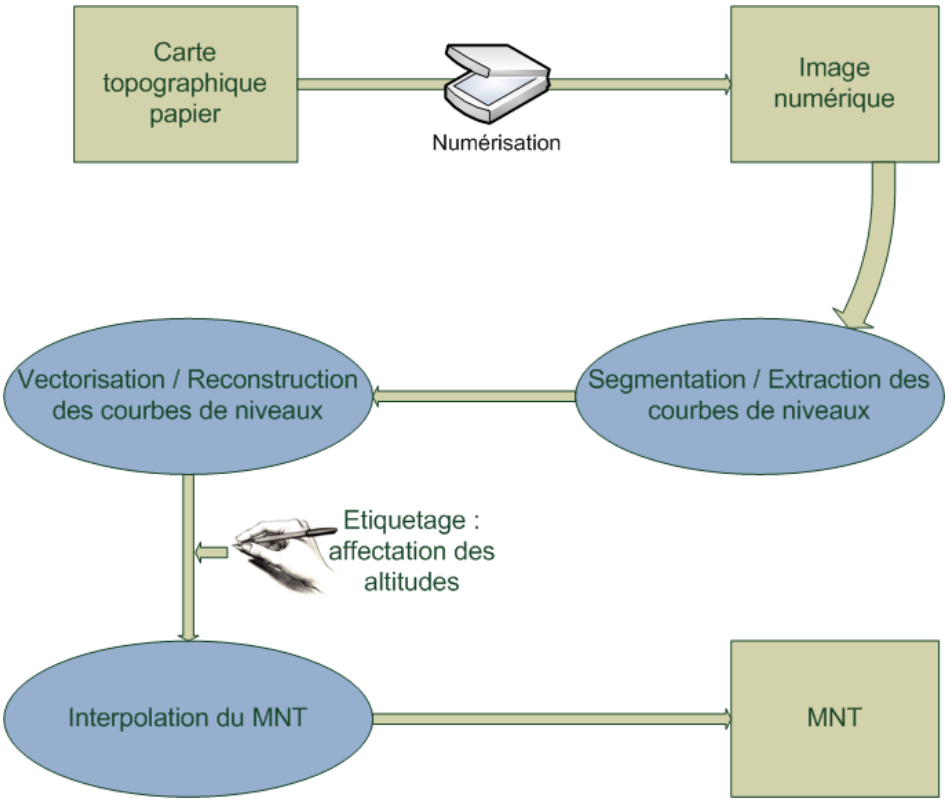


FIG. 2: *Processus de création d'un MNT à partir d'une carte topographique.*

Dans le chapitre 2, nous décrivons différentes techniques classiques permettant de segmenter une carte topographique couleur afin d'en extraire les pixels constituant les courbes de niveau.

Nous décrivons ensuite, dans le chapitre 3 le problème de la reconstruction des courbes interrompues en détaillant une technique efficace que nous avons mise au point pour répondre à ce problème. Nous y présentons également succinctement le problème de la cotation semi-automatique des courbes de niveau.

Le chapitre 4 s'intéresse au point 4 et d'une manière générale de la création d'un MNT à partir de données altimétriques échantillonnées par des courbes de niveau ou des valeurs éparses. Nous présentons dans ce chapitre notre technique d'interpolation hiérarchique lisse à l'aide de fonctions de bases radiales.

Enfin, le chapitre 5 présente le logiciel `AutoDEM` que nous avons développé durant cette thèse. Il intègre l'ensemble des quatre grandes parties décrites dans ces premiers chapitres.

Chapitre 1

Définitions

Le présent chapitre constitue une brève introduction à la thématique de l'informatique appliquée à la géographie que l'on nomme *géomatique*. Nous présentons les concepts de base des systèmes d'information géographique ainsi que les différents objets géographiques manipulés dans cette thèse : courbes de niveau, cartes topographiques et modèles de terrains.

1.1 La géomatique

La géomatique est une discipline qui s'intéresse à associer l'information avec un territoire géographique donné. Plus précisément, elle associe différentes disciplines et techniques telles que la cartographie, la topographie, les systèmes d'information géographiques, l'imagerie aérienne, les techniques de positionnement (par GPS par exemple), etc. Son but est de permettre la représentation, l'analyse et l'intégration des données géographiques. La géomatique repose donc sur trois grandes activités distinctes :

la collecte qui consister à acquérir une représentation numérique du territoire sous la forme de relevés effectués sur le terrain – *points levés* par GPS ou autre appareil de mesure – d'une image satellite, d'une photographie aérienne, de documents cartographiques papiers – plan cadastral ou carte topographique par exemple.

le traitement qui s'effectue souvent avec des logiciels appelés SIG (systèmes d'information géographiques) ;

la diffusion pouvant se faire sur différents médias : cartes topographiques, cartes interactives pour terminaux légers, navigateurs Internet, etc.

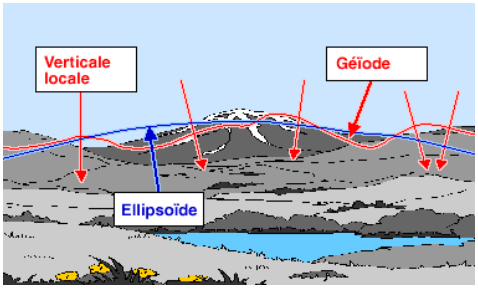
1.2 Modélisation géographique

1.2.0.1 Modélisation de la Terre

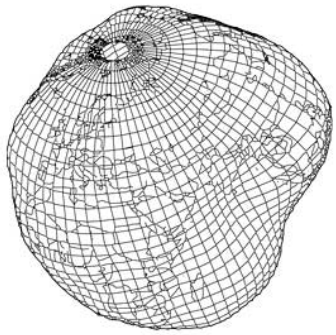
Le *géoïde* est la surface équipotentielle du champ de pesanteur terrestre qui coïncide au mieux avec le niveau moyen des mers (voir figure 1.1). Le géoïde est donc un modèle physique de la Terre décrivant son champ de gravité. La surface de ce géoïde étant irrégulière, sa modélisation est complexe.

L'*ellipsoïde géodésique* est une surface mathématique simple approximant le géoïde. Il est généralement caractérisée par son demi-grand axe a et un coefficient d'aplatissement f. Un

grand nombre d'ellipsoïdes géodésiques ont été calculés en fonction d'hypothèses diverses et de conditions locales propres à chaque pays. Le repère affine dans lequel un ellipsoïde est défini est appelé *référentiel géodésique*. L'ellipsoïde le plus utilisé pour modéliser la Terre est actuellement celui défini par le système géodésique WGS 84 (*World Geodesic System of 84*). Ses caractéristiques sont : $a = 6378137m$ et $f = 1/298,25722357$.



(a) Géoïde et ellipsoïde terrestre.



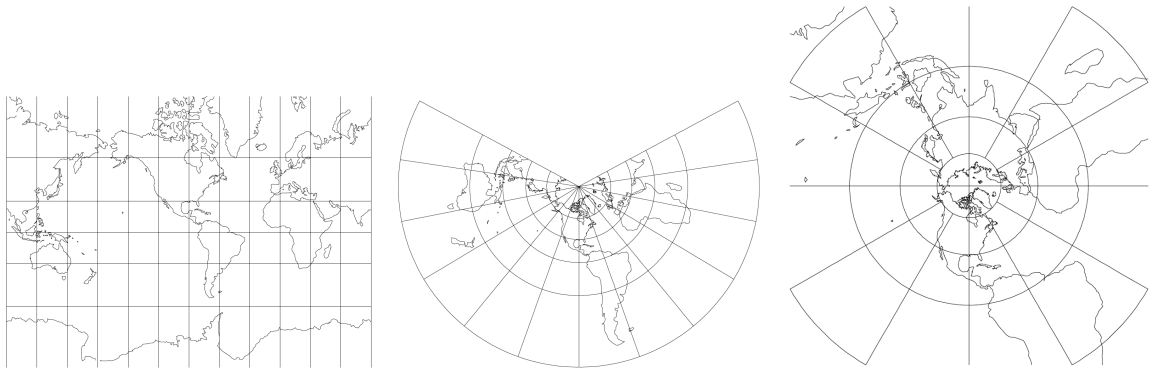
(b) Vue 3D du géoïde de la Terre exagérée 15000 fois.

FIG. 1.1: *La géoïde Terrestre.*

1.2.1 Coordonnées et projections

FIG. 1.2: *Mappemonde de la Géographie de Ptolémée - Edition de Bâle de 1545 (in H. KRAEMER, L'Univers et l'Humanité, vol III, Bong et Cie, Paris, 190X*

Les projections cartographiques sont des techniques permettant de représenter la surface ellipsoïdale de la Terre sur la surface plane d'une carte. C'est l'astronome grec Claudius Ptolémée (100-170) qui, dans son ouvrage *Geographia* [Pto89], introduisit le premier une méthode pour projeter la sphère Terrestre sur une surface plane à l'aide d'un système de *coordonnées* dites *géographiques* basées sur des lignes de *latitude* et de *longitude*. La figure 1.2 présente la carte produite par Ptolémée représentant le monde tel qu'il était connu des Grecs à cette époque.



(a) Projection Mercator. (b) Projection de Lambert conforme. (c) Projection stéréographique polaire.

FIG. 1.3: *Projections cartographiques de la Terre.*

Tout point de la surface peut donc se projeter sur un ellipsoïde géodésique et défini par sa latitude : angle orienté dans le méridien du point entre l'Equateur et la normale à l'ellipsoïde en ce point ; et sa longitude : angle orienté entre le plan méridien d'origine (Greenwich, par convention) et le plan méridien du point en question. L'*altitude* du point représente quant à lui sa distance au niveau moyen de la mer et donc au géoïde.

En cartographie, les coordonnées géographiques sont transformées en coordonnées rectangulaires planes dans un repère affiné à deux dimensions via une *projection cartographique*.

Il existe 3 principaux types de projection cartographique :

- la projection cylindrique par exemple les projections de Mercator, Peters, Robinson ou UTM ;
- la projection conique telle que la projection de Lambert ;
- la projection azimutale dont fait partie la projection stéréographique polaire.

Chaque type de projection a ses avantages et ses inconvénients. En effet, de par les déformations inévitables qu'elles induisent, l'inconvénient principal est qu'une projection cartographique ne permet jamais un résultat exact en toute circonstance. Aussi chaque type de projection doit être utilisé en fonction d'un usage précis. Les projections sont ainsi classées selon les propriétés qu'elles conservent :

- Les projections *équivalentes* conservent les surfaces.
- Les projections *conformes* conservent localement les angles, donc les formes.
- Les projections *équidistantes* conservent les distances sur les méridiens.
- Les autres projections, qui ne conservent ni les angles ni les surfaces, sont dites *aphylactiques*.

Toutes ces projections peuvent être vues comme des fonctions qui transforment une partie de la surface d'une sphère en un domaine planaire. La transformation inverse est habituellement appelée *paramétrisation*. Nous renvoyons le lecteur à [FH05] pour un bon aperçu des techniques de paramétrisation.

1.3 Les systèmes d'information géographique

1.3.1 Définition

Un SIG permet de gérer des données alpha numériques spatialement localisées ainsi que les données graphiques permettant d'afficher ou d'imprimer plans et cartes. Ses usages couvrent les activités géomatiques de traitement et diffusion de l'information géographique.

Le rôle du système d'information est de proposer une représentation plus ou moins réaliste de l'environnement spatial en se basant sur des primitives graphiques telles que des points, des arcs, des polygones ou des images. A ces primitives sont associées des informations qualitatives telles que leur nature (par exemple : route, voie ferrée, forêt, etc.) ou toute autre information contextuelle.

L'information géographique peut être définie comme l'ensemble de la description d'un objet et de sa position géographique à la surface de la Terre.

Le développement des SIG est étroitement lié à celui de l'informatique. Longley *et al.* [LGMR99] distinguent trois périodes dans cette évolution :

1950-1970 : avec l'apparition de l'informatique, les premières applications de cartographie assistées par ordinateur voient le jour ;

1970-1980 : diffusion des outils de cartographie automatique et des SIG dans les organismes d'État tels que le cadastre, l'armée, etc. ;

1980-2007 : croissance du marché des SIG, développement d'applications performantes sur PC, et mise en réseau des informations aboutissant aux applications sur Internet.

Depuis quelques années, on assiste à la diffusion vers le grand public de l'usage de l'information géographique avec en particulier les solutions embarquées sur GPS, le calcul d'itinéraires routiers et la cartographie sur Internet.

1.3.1.1 Les 5 fonctions

On résume souvent les SIG aux 5 fonctions suivantes, parfois regroupées sous le terme des 5A :

Abstraire : modélisation de l'information et définition des données ;

Acquérir : saisie des informations géographiques sous forme numérique ;

Archivr :stockage des données afin de pouvoir y accéder facilement ;

Analyser : manipulation et interrogation des données géographiques ;

Affichager : restitution de l'information géographique sous forme graphique.

1.3.1.2 Les données

Les données géographiques sont constituées de quatre composantes :
- les données géométriques renvoient à la forme et à la localisation des objets ou phénomènes ;
- les données descriptives renvoient à l'ensemble des attributs descriptifs des objets et phénomènes à l'exception de la forme et de la localisation ;
- les données graphiques renvoient aux paramètres d'affichage des objets (type de trait, couleur, etc.) ;
- les méta-données associées, c'est-à-dire les informations associées aux données (date d'acquisition, nom du propriétaire, méthodes d'acquisition, etc.)

1.3.2 Les couches

Dans un SIG, les objets géographiques sont organisés en couches (voir figure 1.4). Généralement, une couche fait référence à un thème particulier et donc à une structure de données particulière. En géomatique, on distingue deux grandes familles de couches : les couches dites *raster* et les couches vecteurs. La figure 1.5 illustre ces deux familles de données.

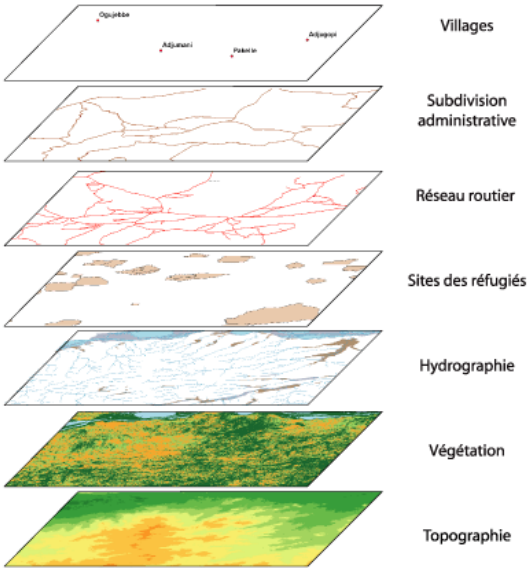


FIG. 1.4: *Exemple de couches d'informations utilisées pour décomposer un territoire.*

1.3.2.1 Couches *raster*

Une couche *raster* (appelées aussi *maillée* ou *tramée*) est une couche dans laquelle les données sont stockées sous forme de tableaux à 2 dimensions auxquels sont associés des informations colorimétriques ou altimétriques. Il s'agit le plus souvent d'une image, d'un plan, d'une photo numérisée ou d'une carte d'élévations. Les couches *raster* sont affichées dans le SIG comme des images.

1.3.2.2 Couches vecteurs

Les objets des couches vecteurs sont représentés par des points, des lignes, des polygones, des polygones à trous ou des chaînes de caractères. Ces couches peuvent contenir par exemple des données décrivant les frontières de zones urbanisées ou forestières, des tracés routiers ou la toponymie d'un lieu.

L'opération consister à convertir une couche raster en une couche vecteur s'appelle la *vectorisation*. L'opération inverse est appelée *rasterisation*.

1.3.3 Logiciels SIG

Il existe des centaines de logiciels SIG. Le plus connu est le logiciel ArcGIS [ArcGIS]. ArcGIS est une suite de logiciels SIG commerciaux développés par la société ESRI. La plate-forme

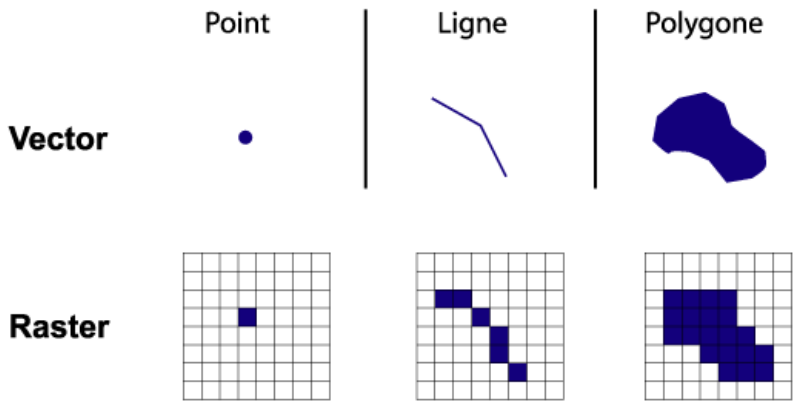


FIG. 1.5: *Représentation d'objets géométriques en mode* raster *et vecteurs.*

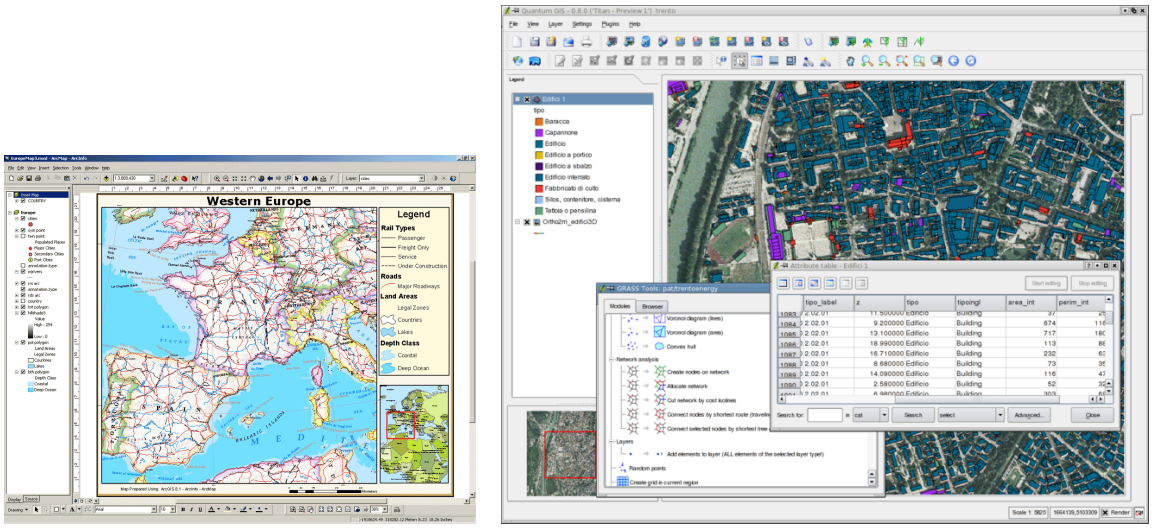


FIG. 1.6: *Capture d'écran des logiciels SIG ArcInfo (à gauche) et GRASS GIS (à droite).*

ArcGIS permet de déployer des fonctionnalités de SIG sur différentes plateformes (bureau, serveur, Web, mobiles). Parmi ces logiciels on compte, entre autres : ArcInfo (voir figure 1.6), ArcEditor, ArcView et ArcReader.

GRASS GIS (*Geographic Resources Analysis Support System*) [GRASS] est un SIG qui a été originellement développé par des ingénieurs de l'*U.S. Army Construction Engineering Research Laboratories* (USA-CERL). Depuis, ce logiciel est maintenu et amélioré par de nombreux programmeurs, et maintenu officiellement par des institutions aux Etats-Unis, en Italie et en Allemagne. Parmi les utilisateurs de GRASS on dénombre de nombreuses organisations américaines telles que la NASA, le U.S. National Park Service ou l'USGS (*United States Geological Survey*), mais également par de nombreuses universités et entreprises de par le monde.

GRASS est un SIG complet développé en C, et offrant, par le biais d'un shell en ligne de commande ou d'une interface utilisateur graphique (voir figure 1.6), des centaines d'outils et programmes pour créer, manipuler et gérer l'information géographique. GRASS est distribué

sous licence GPL et disponible pour les principales architectures et systèmes d'exploitation (Unix/Linux, MacOS et MS Windows).

Citons enfin MapServer [MapServer], un environnement de développement Open Source permettant de construire des applications Internet géoréférencées très en vogue ces dernières années. MapServer n'a pas vocation à être un SIG complet mais a pour but d'afficher des données spatiales sur le web (cartes, images, et données vectorielles). MapServer a été développé par l'Université du Minnesota (UMN) en coopération avec la NASA et le Minnesota Department of Natural Resources (MNDNR).

1.4 Les cartes topographiques

La topographie est l'art de la mesure puis de la représentation sur un plan ou une carte des formes et détails visibles sur le terrain, qu'ils soient naturels, tel que le relief, ou artificiels tels que les routes ou bâtiments. Son objectif est de déterminer la position et l'altitude de n'importe quel point situé dans une zone donnée qu'elle soit de la taille d'un continent ou d'un champ.

Une carte topographique est une carte représentant la topographie d'une région géographique donnée de manière précise et détaillée. Sur une telle carte, le relief du terrain est représenté sous la forme de courbes de niveau ou de points cotés.

1.5 Modélisation topographique de terrains

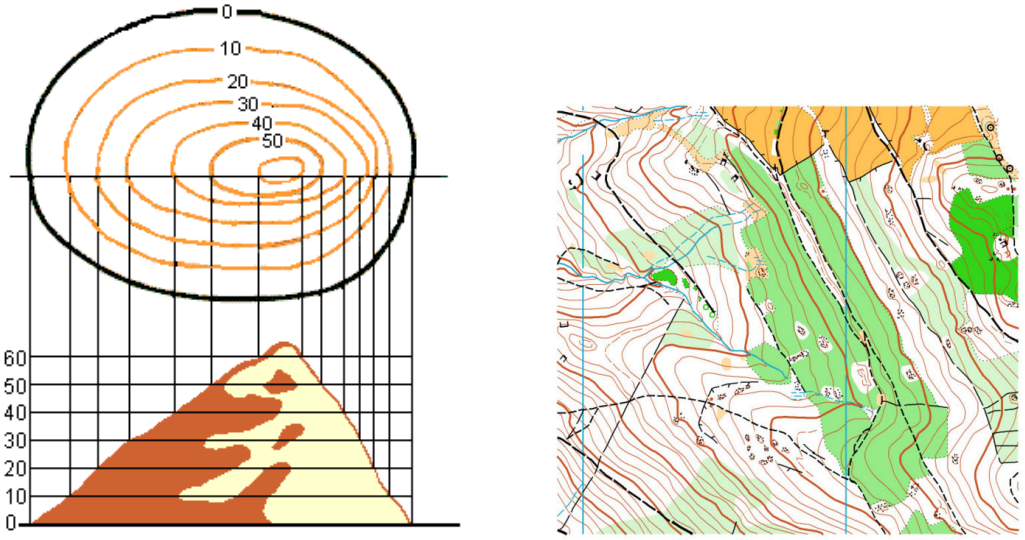
Le terme modèle numérique de terrain (MNT, en anglais DTM) désigne une représentation de la topographie (ou altimétrie) d'une zone géographique adaptée aux traitements informatiques. Sur les cartes topographiques, la topographie du terrain est représentée à l'aide de *courbes de niveau*. Cette représentation donne une indication importante à l'observateur humain, mais n'est pas adaptée à un traitement informatique. En effet, l'évaluation de l'altitude en un point donné n'est pas immédiate comme nous le verrons dans le chapitre 4. Il existe de différentes manières de modéliser la topographie en géomatique. Nous présentons les plus utilisés : les cartes d'élévations et les modèles triangulés.

1.5.1 Courbes de niveau

Les courbes de niveau sont des lignes imaginaires placées sur une carte de géographie, qui joignent tous les points situés à la même altitude. C'est aussi la ligne d'intersection d'un plan horizontal avec le relief du terrain (voir figure 1.7). Plus les courbes de niveau sont rapprochées, plus la pente est forte. Entre deux courbes de niveau successives, on admet que la pente est régulière. La distance verticale qui sépare deux courbes de niveau successives s'appelle *l'équidistance*.

Sur une carte, on peut distinguer 3 types de courbes de niveau :

- les courbes *directrices*, principales ou maîtresses dessinées avec un trait épais continu et repérées par une indication d'altitude ;
- les courbes "*traditionnelles*" dessinées en trait fin continu ;
- les courbes *intermédiaires*, dessinées en traitillés ou en pointillés sur la carte et qui se situent à la demi-équidistance. On les représente sur la carte uniquement lorsque la pente n'est pas régulière entre deux courbes de niveau "traditionnelles" ou entre une courbe directrice et une courbe "traditionnelle".



(a) Echantillonnage des courbes de niveau.

(b) Courbes de niveau imprimées en orange sur une carte topographique couleur. Les courbes plus épaisses sont les courbes dites *directrices*.

FIG. 1.7: *Construction et représentation des courbes de niveaux.*

Dans les SIG, les courbes de niveau peuvent être stockées de 2 façons. La plus efficace est la description sous forme vectorielle. Une courbe de niveau est alors assimilée à une polyligne ou une courbe spline. La manipulation de tels objets est alors aisée : il est facile de supprimer une courbe ou un point de celle-ci. Il existe différents formats pour stocker les courbes vectorielles. L'USGS les diffuse par exemple sous la forme de fichier DLG (*Digital Line Graphs*). Un autre moyen de stockage consiste à les décrire sur une couche *raster*. Les courbes sont alors dessinées sur l'image d'une carte topographique 1.8 ou dans une carte d'élévation. Dans ce cas, la manipulation des courbes se fait par modification des pixels de l'image. Pour pouvoir effectuer des traitements sur les courbes, il est alors souvent nécessaire de les vectoriser.

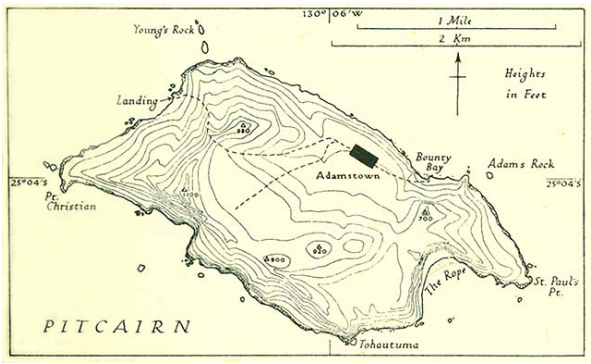


FIG. 1.8: *Carte topographique ancienne de l'île Pitcairn dans l'Océan Pacifique.*

1.5.2 Les cartes d'élévation

Une carte d'élévation (appelé aussi MNA : modèle numérique d'altitude ou MNE : modèle numérique d'élévation) est une grille régulière contenant des valeurs d'altitudes échantillonnées ou interpolées et uniformément espacées selon la définition de Burrough [Bur86]. La grille est stockée sous la forme d'un tableau 2D dont les paramètres sont la résolution dans les directions X et Y. Ces valeurs doivent être choisies de façon à permettre la représentation d'une grande diversité de régions : dans le cas d'une zone montagneuse, une grande résolution permettra de mieux représenter les aléas du terrain, tandis que pour une plaine, une résolution moindre suffira. Une estimation de l'altitude entre 4 points voisins de la grille pourra être obtenue à l'aide d'un schéma d'interpolation (bilinéaire, bicubique, etc.) Il est évident qu'une résolution plus précise requiert non seulement une quantité de données supplémentaires, mais aussi un temps de calcul plus long. Il est donc important de trouver un compromis entre la précision des données désirée, et la taille mémoire et le temps de calcul disponibles.

La figure 1.9 montre un MNA affiché sous forme d'image en niveaux de gris et en couleur avec une palette topographique.

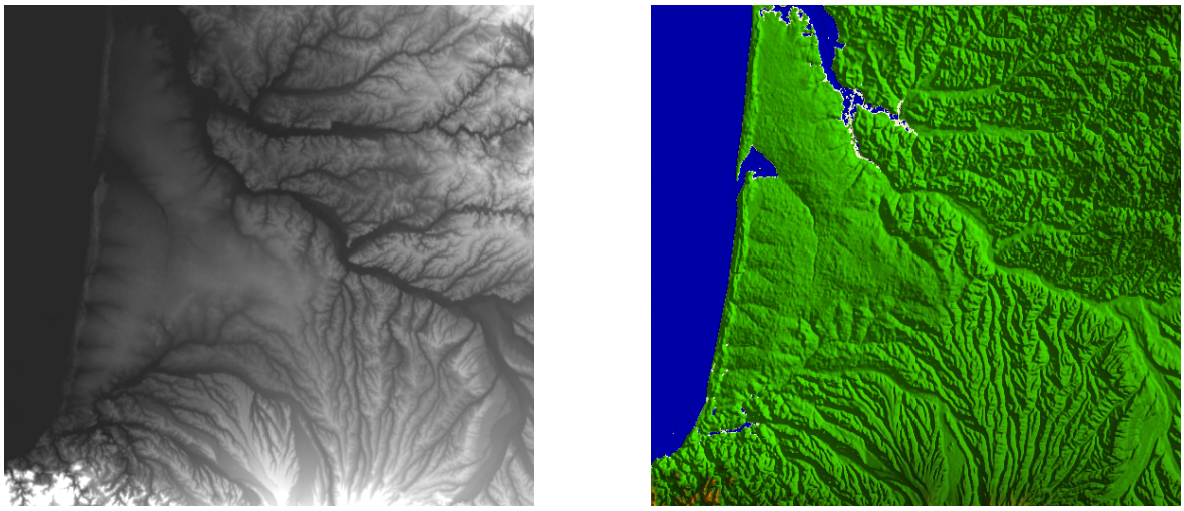


FIG. 1.9: *Carte d'élévation de la région Aquitaine affichée sous forme d'image en niveaux de gris (à gauche) et en couleur avec un effet d'ombrage (à droite).*

Il est à noter que si les MNA décrivent une surface en 3 dimensions, la surface représentée est dite de dimension 2,5 puisque chaque point de la surface ne peut être associé qu'à une et une seule valeur d'altitude. Cette particularité empêche donc de représenter des spécificités de la surface telles que des cavités ou promontoires naturels.

1.5.3 Les modèles triangulés

Pour représenter un terrain, une autre alternative efficace aux cartes d'élévation denses a été introduite en cartographie par Peucker *et al.* [PC75] et s'appelle Réseaux de Triangles Irréguliers (RTI, en anglais TIN : *Triangulated Irregular Network*).

Les RTI représentent la surface à l'aide d'un ensemble de points irrégulièrement espacés reliés entre eux pour former un réseau de triangles. Des points d'élévation irrégulièrement

espacés sont sélectionnés pour représenter le terrain avec généralement un grand nombre de points dans les régions accidentées et un petit nombre de points dans les régions relativement planes.

Il existe différentes stratégies de création de RTI et nous en détaillerons quelques-unes dans le chapitre 4. D'une manière générale, le RTI est obtenu par triangulation de Delaunay. La surface triangulée définie par des facettes de triangles de taille et forme irrégulières ne se chevauchant pas est donc continue. A l'intérieur de chaque triangle, la surface est supposée homogène et représente un plan orienté dans l'espace.

Les arêtes du RTI peuvent permettre de capturer la position de certaines caractéristiques géomorphologiques de la surface jouant un rôle important comme les lignes de crêtes (en anglais *ridge lines*, lignes de points hauts d'un relief séparant deux versants opposés aussi appelées lignes de partage des eaux, dorsales ou interfluves) ou les lignes de Talweg (lignes reliant les points les plus bas d'une vallée, aussi appelées lignes de collecte des eaux). La figure 1.5.3 montre 3 courbes de niveau et les points et facettes d'un modèle RTI. La figure 1.5.3 illustre ce RTI à l'aide d'une vue 3D.

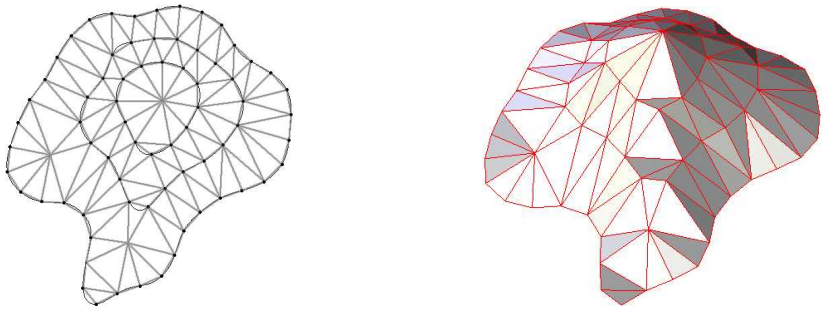


FIG. 1.10: *Exemple d'un modèle RTI et sa vue en 3D.*

Il existe de nombreux autres types de modèles triangulés utilisés particulièrement pour la visualisation interactive de terrain. Ces modèles reposent généralement sur des structures hiérarchiques (par exemple le modèle QTM, *Quaternary Terrain Model* de Dutton [Dut84]) permettant de disposer d'un modèle multi-résolution et d'adapter la quantité de données utilisées en fonction de différents critères tels que la distance au point de vue. Nous décrirons plus en détail ces mécanismes dans la deuxième partie de cette thèse.

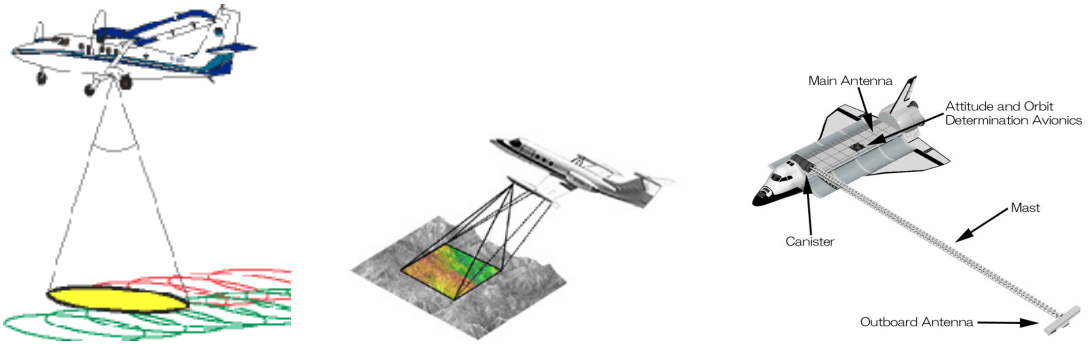
En conclusion, il n'y a pas de modèle meilleur dans tous les cas. Le choix d'un modèle peut s'effectuer en fonction de différents critères tels que :

- la disponibilité des données ;
- la nature de la surface (plane ou montagneuse) ;
- l'application, en fonction des techniques qui seront utilisées pour analyser, manipuler ou visualiser le modèle ;
- l'échelle et la résolution des données.

1.6 Quelques techniques d'acquisition topographiques

Aujourd'hui, les données topographiques peuvent être acquises à l'aide de diverses technologies d'acquisition à distance (en anglais, *remote sensing*). La plupart de ces techniques font usage de radiations électromagnétiques émises ou réfléchies de l'objet d'intérêt dans un certain domaine fréquentiel (infrarouge, lumière visible, micro-ondes, etc.) Cela est rendu pos-

sible par le fait que les objets examinés (plantes, maisons, rivières, masses d'air, etc.) reflètent ou émettent dans différentes longueurs d'onde et selon différentes intensités. Nous présentons maintenant deux des technologies les plus utilisées pour la télé-acquisition topographique.



(a) Laser LiDAR (b) IfSAR (c) La navette Endeavour de la mission SRTM

FIG. 1.11: *Illustration des techniques d'échantillonnage de la Terre.*

1.6.1 LiDAR

La technologie LiDAR (*Light Detection and Ranging*) est basée sur une sonde active, similaire au radar, qui transmet des impulsions laser sur une cible et enregistre le temps entre l'émission et la réception sur le récepteur de la sonde. Cette technologie est utilisée pour échantillonner la surface terrestre en haute résolution en montant une sonde LiDAR sous un avion et en l'associant à un récepteur GPS et à une centrale inertielle (appareil permettant de calculer l'évolution du vecteur vitesse et de la position, ainsi que son orientation à l'aide de différents capteurs). On mesure alors le temps de retour de l'impulsion pour déterminer l'altitude de la surface (voir figure 1.12). Les données obtenues ont la forme d'un nuage de points. Cette technologie a été développée à la fin des années 1960, mais la première application commerciale n'a eu lieu qu'en 1993.

1.6.2 IfSAR

La technologie IfSAR ou InSAR (*Interferometric Synthetic Aperture Radar*) est basée sur une sonde radar généralement montée sur un avion et utilisée pour mesurer la topographie d'une surface. Les impulsions Radar pulses sont dirigées sur la surface terrestre et les signaux captés en retour à l'aide d'antennes permettent de déterminer l'altitude du point visé sur la surface. Les coordonnées au sol sont, comme dans la technologie LiDAR, déterminées au moyen d'un capteur GPS et d'une centrale inertielle. Bien que développée dans les années 1960 cette technologie ne sera utilisée pour pour effectuer des mesures topographiques que dans les années 1970 et la première utilisation commerciale ne se fera qu'en 1996.

C'est lors de la mission SRTM (*Shuttle Radar Topography Mission*) menée par le *National Geospatial-Intelligence Agency* (NGA) et la *National Aeronautics and Space Administration* (NASA) que la technologie IfSAR sera mise en avant. En février 2000 et durant 11 jours, la navette spatiale Endeavour a recueilli des données altimétriques pour plus de 80% des terres émergées. Trois types de MNA ont été dérivés de cette mission :

– SRTM-1 d'une résolution de 30m mais couvrant uniquement les USA ;

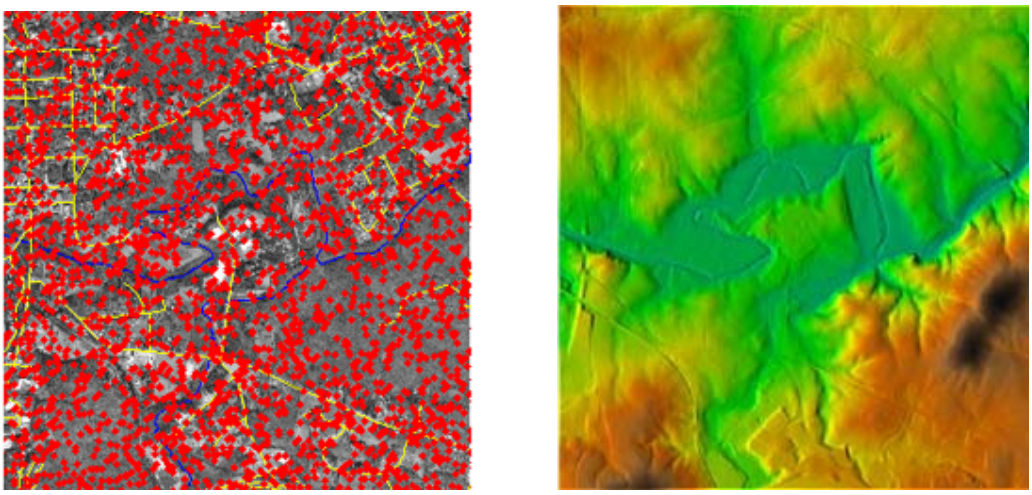


FIG. 1.12: *Acquisition par la technologie LiDAR. A gauche les échantillons capturés sont en rouge (pour plus de clarté, seul 1 point pour 1000 est affiché). A droite le MNA reconstruit à partir de ces échantillons, la couleur étant relative à l'altitude.*

	LiDAR	IfSAR P-band
Taux d'impulsion	$\leq$ 40 KHz	
Longueur d'onde	1045 - 1065 μm (quasi infrarouge)	
Altitude	300 - 2000 m	Jusqu'à 0,70 x altitude (mètres)
RMSE Précision en Z (verticale)	env. 15 cm*	1 - 5 m
RMSE RMSE X, Y (horizontale)	$\leq$ 1 m*	2 - 4 m
Résolution (espacement des points)	$\geq$ 0.75 m	2,5 ; 5 ou 10 m

TAB. 1.1: *Caractéristiques de différentes techniques d'acquisition topographiques.*

– SRTM-3 d'une résolution de 90m (80% des terres) ;
– SRTM-30 d'une résolution de 1km couvrant la surface de la planète.

Les données issues de cette mission sont disponibles publiquement ou sous forme payante pour les plus précises sur le site de l'USGS.

Le tableau 1.1 récapitule les caractéristiques techniques des 2 technologies LiDAR et IfSAR.

Chapitre 2

Segmentation des cartes topographiques

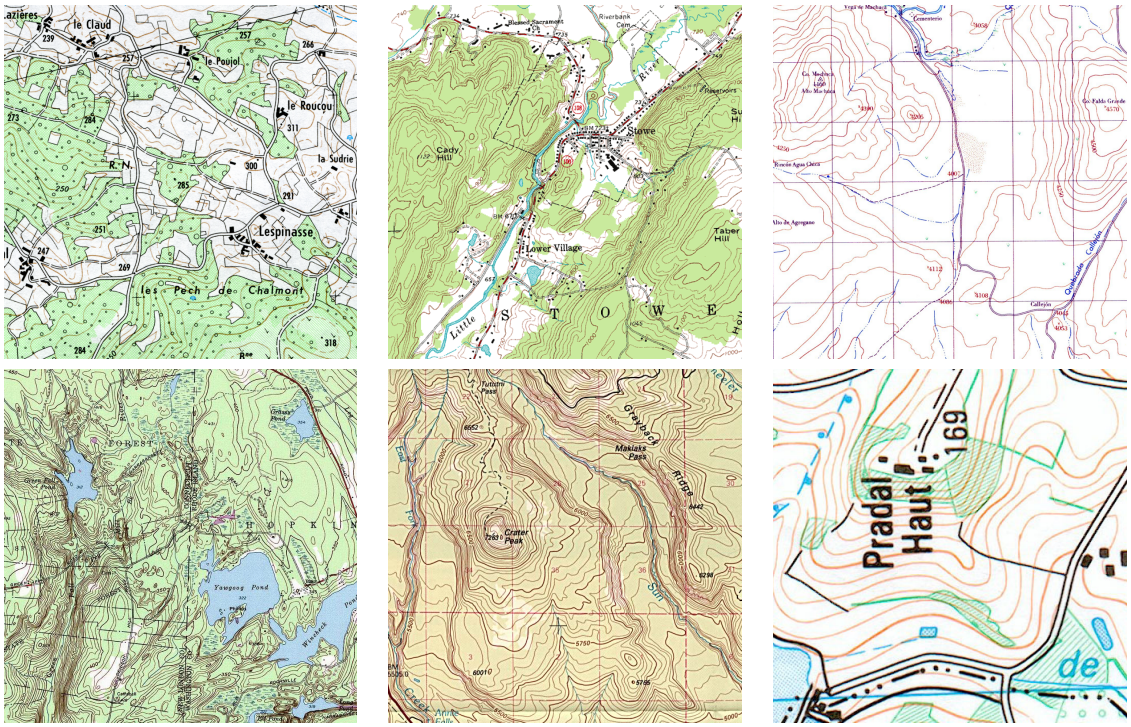


Fig. 2.1: *Extraits de cartes topographiques numérisées.*

2.1 Introduction

Depuis des décennies, l'analyse automatisée de cartes topographiques a été l'objet, de nombreuses publications. Des travaux anciens sur la vectorisation de lignes dessinées avait introduit les différentes étapes principales nécessaires pour une telle procédure automatisée :

1. numérisation du document papier ;

2. filtrage ;
3. seuillage ;
4. squelettisation et ébarbulage de l'image binaire ;
5. vectorisation des composantes objets.

Ces étapes sont décrites dans [LO82] pour la vectorisation automatique de courbes de niveau ou des cours d'eau dessinés sur des feuilles, et dans [Gre87] pour l'extraction de courbes de niveau sur des cartes topographiques numérisées en noir et blanc.

Depuis, nombreux travaux se sont intéressés à ce même problème sur des cartes topographiques couleur numérisées. Dans ce chapitre nous décrivons différentes méthodes existantes et permettant l'extraction des courbes de niveau sur ce type de cartes.

S'il existe un grand nombre de cartes topographiques de par le monde, il n'existe cependant pas de standard en matière d'écriture de ces cartes, et chaque éditeur peut utiliser ses propres règles d'édition. Généralement, les cartes topographiques contiennent, outre les courbes de niveau, de nombreuses couches informations thématiques telles que les routes, les cours d'eau, les bâtiments, les forêts, les toponymes, etc. Le plus souvent, chaque type d'information est caractérisé par une couleur spécifique. Lors de l'impression de la carte, les différentes couches sont mixées dans un certain ordre.

Sur la plupart des cartes topographiques en couleur, les courbes de niveau sont tracées en marron/orange (cf. figure 2.1). L'extraction des courbes va donc consister en une segmentation de l'image sur des critères colorimétriques ponctuels ou locaux, en y adjoignant éventuellement des contraintes de voisinage. Dans tous les cas, qu'elle passe ou non par une étape de classification, l'étape d'extraction doit aboutir à une carte binaire filtrée décrivant l'appartenance potentielle d'un point à une courbe ou non. C'est à partir de cette carte que l'étape de reconstruction s'attachera à vectoriser les objets discrets obtenus.

Cependant, cette étape n'est pas forcément aisée et Kothenzad [KZ03] identifie quatre challenges principaux dans l'extraction des courbes de niveau :

- *l'effet de crènelage* (*aliasing* en anglais) induit par la fonction du transfert de points. Par exemple, la numérisation d'une ligne noire sur un fond blanc entraînera un lissage à la frontière d'une zone blanche et d'une zone noire ainsi que la présence d'une multitude de niveaux de gris ;
- *la proximité* des différents objets : s'ils sont normalement séparés par la ou les couleurs du fond de la carte, cet espace peut être érodé par l'effet de crènelage et donc il peut devenir difficile de séparer les différentes composantes de cette manière (voir figure 2.2).
- l'existence de *fausses couleurs* dues au mauvais étalonnage RGB du scanner. Cette erreur peut être physique si les capteurs ne sont pas parfaitement alignés ou optique à cause de l'effet de prisme de la lentille (voir figure 2.3) entraînant des erreurs dites de distorsion ou d'aberrations chromatiques.
- *l'occultation* et *l'intersection* des différents objets linéaires sur la carte, par exemple lorsqu'une courbe de niveau intersecte le tracé d'un cours d'eau.

A cela, on peut ajouter également le problème de la qualité d'impression de la carte papier. Par exemple, des problèmes de calage peuvent survenir entre les couches de couleur cyan, magenta et jaune utilisées lors de l'impression de la carte couleur. Comme l'illustre la figure 2.4 un tel décalage peut entraîner l'apparition de lignes pouvant être assimilées à des morceaux de courbes.

L'étape de la numérisation est donc cruciale pour la suite du traitement et doit être effectuée avec soin. L'ensemble du processus de reconstruction d'un MNT reposant sur le

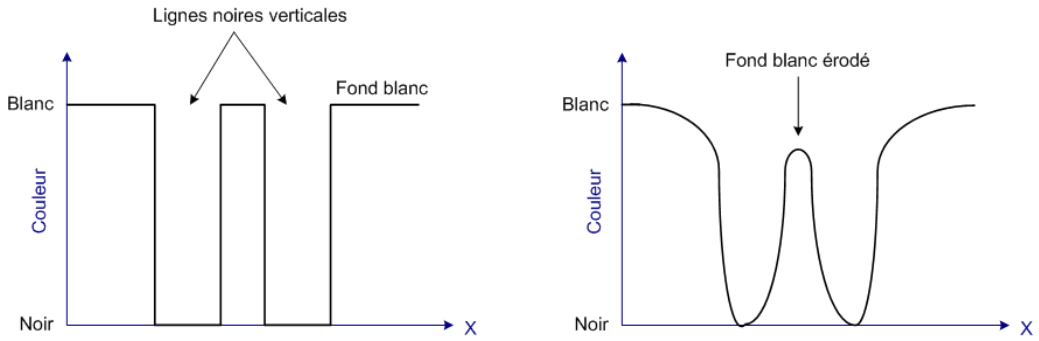


FIG. 2.2: *Effet de crènelage.*

traitement des courbes de niveau de la carte, il est capital que celles-ci répondent aux critères suivants sur l'image numérisée :

- les courbes de niveau doivent être suffisamment épaisses,
- elles doivent être le plus isolées possible les unes des autres,
- la couleur qui les caractérise doit être la plus uniforme possible

Ces réglages doivent donc être effectués en fonction du scanner utilisé, de la qualité de la carte, de son type (topographique seulement, mixte...) ainsi que de son échelle. La résolution de la numérisation doit donc être suffisante. Différents essais ont montré qu'une résolution minimale de 400 ou 500dpi est suffisante.

Dans les sections suivantes, nous présentons différentes méthodes de segmentation pouvant être utilisées pour analyser les cartes topographiques. Ces méthodes reposent toujours sur l'analyse des composantes couleur des pixels de l'image dans le but d'en réaliser un étiquetage. Le choix du modèle de couleur est un critère important. L'espace RGB est le plus utilisé pour décrire les images numériques, car il est utilisé pour l'affichage des écrans. Cependant, les principaux inconvénients de ce modèle sont que les composantes sont fortement corrélées, que son interprétation humaine n'est pas évidente et qu'il n'est pas uniforme du point de vue perceptif. L'espace TSV (teinte-saturation-valeur, en anglais *hue-saturation-value*, HSV) est un modèle adapté à l'interface homme-machine. Cet espace est souvent utilisé en segmentation d'images couleur, mais sa supériorité sur l'espace RGB reste controversée. L'espace couleur CIE L*a*b* dit aussi CIELAB (L* correspond à la luminance, a* et b* sont les chromacités des couples antagonistes vert-rouge et bleu-jaune respectivement) est la représentation la plus

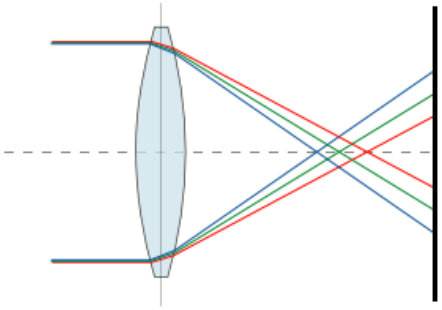


FIG. 2.3: *Aberration chromatique de la lentille.*

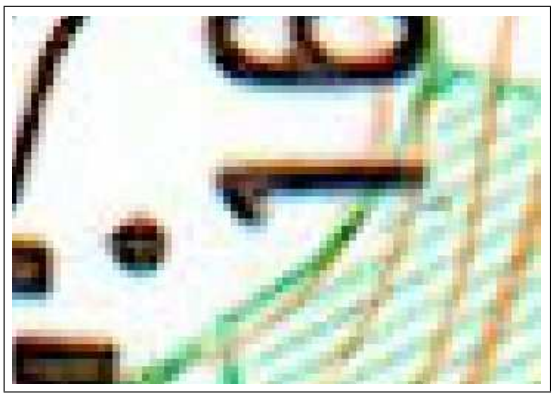


FIG. 2.4: *Agrandissement d'une zone où l'on distingue des pixels orange à proximité des pixels noirs.*

utilisée en colorimétrie. L'avantage principal de cet espace est qu'il est *perceptuellement uniforme* c'est-à-dire que des différences de couleur perçues comme égales pour l'oeil humain correspondent à des distances Euclidiennes égales. Cependant, la transformation de l'espace RGB vers l'espace CIELAB est complexe et nécessite un passage par l'espace XYZ. Or pour pouvoir spécifier complètement la transformation RGB vers XYZ, il faut connaître les coordonnées des stimulus primaires et le blanc de référence de l'illuminant. Dans la plupart des situations, il faut faire des hypothèses puisque ces informations ne sont pas accessibles.

2.2 Classification colorimétrique

La technique de classification la plus simple est le seuillage. Un seuillage consiste à effectuer un classement binaire des pixels en fonction d'une ou plusieurs valeurs de leurs composantes colorimétriques. Le seuil peut être déterminé automatiquement par analyse de l'histogramme ou manuellement en sélectionnant une plage de couleur représentative de l'objet cartographique à extraire.

Desseiligny et Dupont [PD94][Dup99][DDG98b] segmentent la carte en plusieurs classes correspondant aux couleurs pures utilisées dans la carte et sélectionnées par l'opérateur. La segmentation robuste s'effectue en deux temps. Une première segmentation s'effectue à l'aide de l'algorithme de ligne de partage des eaux sur l'histogramme RGB, celle-ci aboutit

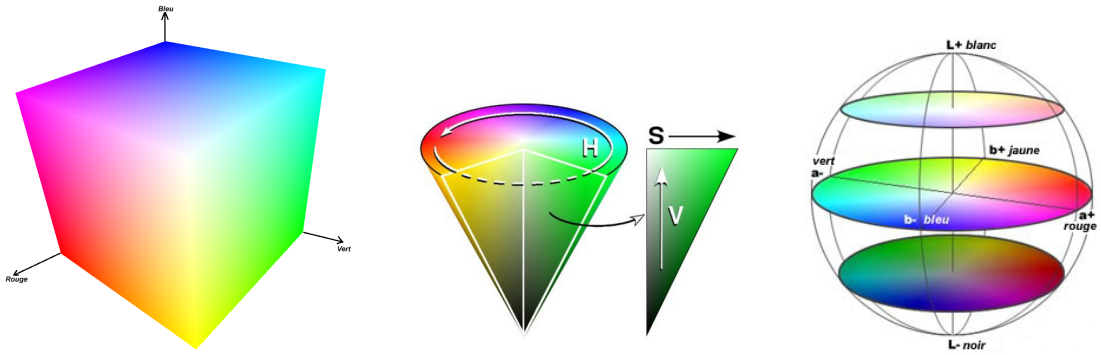


FIG. 2.5: *Représentations volumiques des modèles de couleurs RGB, HSV et CIELAB.*

généralement à une surclassification. Dans un deuxième temps, la surclassification obtenue est résolue par une phase d'affectation supervisée de certaines classes suivie d'une phase de complétion automatique par affectation au plus proche voisin. Cette méthode donne de bons résultats sur les cartes utilisées, mais nécessite l'expertise de l'opérateur.

Arrighi *et al.* [AS99]) s'intéressent uniquement aux pixels constituant les courbes de niveau. Ils procèdent simplement à un seuillage en sélectionnant les pixels dont la *teinte* rouge. Aucune précision n'est cependant donnée sur le seuil utilisé.

Spinello *et al.* [SG04] utilisent quant à eux l'espace TSV. Dans un premier temps, les pixels avec $V < 0.25$ sont classés comme noirs tandis que les pixels avec $S < 0.20$ et $V > 0.60$ sont classés comme blancs. L'histogramme de la composante de teinte T des pixels restants est construit et le pic entre $10 < T < 30$ est considéré

Dans la pratique cette phase doit être interactive. En effet, il apparaît souvent nécessaire que l'opérateur manipule les paramètres du seuillage afin de les adapter au mieux à la carte à traiter.

Les inconvénients sont d'une part une très forte sensibilité au bruit auquel on peut répondre par un simple filtrage, mais surtout si cette technique fonctionne bien avec des cartes simples elle donne de très mauvais résultats avec des cartes plus garnies en informations où les courbes de niveau ont des couleurs très dispersées (notamment si le fond de la carte n'est pas blanc ou si les courbes croisent d'autres symboles comme une zone de hachures vertes comme sur la figure 2.4).

2.2.1 Classification par k-means

La méthode des *k-means* est un outil classique de classification non supervisée permettant de répartir un ensemble de données en k classes homogènes. Cette méthode fut introduite par MacQueen [Mac67] en 1967 et a donné lieu à de nombreuses améliorations.

Dans le cadre de la classification non supervisée, on cherche à partitionner l'espace en classes concentrées et isolées les unes des autres. Dans cette optique, l'algorithme des *k-means* vise à minimiser la variance intra-classe, qui se traduit par la minimisation de l'énergie suivante :

$$E = \frac{1}{2} \sum_{c \in \mathcal{C}} \sum_{x \in c} ||x - m_c||^2 \tag{2.1}$$

avec $\mathcal{C}$ l'ensemble des classes et pour chaque classe c, m_c son centre (appelé noyau), $V(c)$ sa variance, $]c$ le nombre de ses éléments et $\mathcal{D} = \bigcup_{c \in \mathcal{C}} c$ l'ensemble des données que l'on cherche à classer.

La minimisation de cette énergie peut se réaliser par une descente de gradient sur les noyaux. Une fois l'initialisation des k noyaux effectuée, on effectue l'algorithme suivant. Jusqu'à stabilisation des noyaux, pour chaque pixel de l'image : i) déterminer le noyau le plus proche de la valeur associée au pixel (niveau de gris ou composantes couleurs) ; ii) réévaluer des noyaux. Pour accélérer la convergence, l'initialisation des noyaux peut se faire avec les couleurs primaires utilisées sur la carte. L'étape i) implique de calculer la distance entre deux couleurs, il est donc conseillé d'utiliser un modèle uniforme tel que le modèle L*a*b*.

FIG. 2.6: *Classification par k-means en 5 classes.*

2.3 Segmentation par apprentissage

Les techniques de segmentation par apprentissage se distinguent des techniques précédentes en ce qu'elles nécessitent l'expertise de l'utilisateur pour leur fournir un jeu de tests et les résultats à atteindre. Plus complexes à utiliser, elles ont cependant l'avantage de pouvoir s'adapter aux spécificités de la carte étudiée et ainsi d'obtenir de meilleurs résultats sur des cartes plus difficiles à analyser. Nous présentons d'abord une technique utilisant des échantillons décrivant les différentes configurations de composition colorimétriques de la carte. Nous présentons ensuite des techniques de segmentation par réseaux de neurones articifiels.

2.3.1 Echantillonnage

Khotanzad *et al.* [KZ96][KZ03] proposent d'utiliser un ensemble d'échantillons combinant différentes couleurs de la carte pour surmonter le problème de l'aliassage et des fausses couleurs décrit plus haut. Les échantillons de taille 16×16 pixels sont sélectionnés par l'opérateur et permettent de constituer une mosaïque de couleurs "solides" ou de combinaisons de couleurs (voir figure 2.7). Un échantillon de couleur solide correspond à une zone uniforme contenant la couleur du fond de carte ou à une région illustrée par un aplat de couleur (par exemple une zone forestière en vert). Un échantillon décrivant une combinaison de couleurs correspond à une région où un objet linéaire recouvre partiellement une couleur solide. La couleur médiane des différents échantillons d'une couleur solide permet de déterminer une couleur de référence de celle-ci. Pour les échantillons de combinaisons de couleurs, l'axe majeur de l'ellipsoïde contenant l'ensemble des couleurs des pixels dans l'espace RGB est calculé par régression linéaire sur vecteur propre. Chaque pixel est ensuite classifié selon un critère de distance minimum entre les vecteurs et les valeurs propres des couleurs de références.

Solid Color Key	Samples
White	
Green	
Red	

Combination Color Key	Samples
Black on white	
Black on green	
Brown on white	
Brown on green	
Blue on white	
Blue on green	
Purple on white	
Purple on green	

FIG. 2.7: *Mosaïque de couleurs solides (à gauche) et de combinaisons de couleurs (à droite).*

2.3.2 Réseaux de neurones artificiels

Un réseau de neurones artificiels est un modèle de calcul dont la conception est très schématiquement inspirée du fonctionnement des neurones biologiques. Le principe du réseau de neurones est de mettre en œuvre un principe d'induction, c'est-à-dire d'apprentissage par expérience. Un tel réseau est constitué de couches de neurones artificiels reliés entre eux. Le neurone artificiel (voir figure 2.8) calcule la somme de ses entrées puis cette valeur passe à travers la fonction d'activation pour produire sa sortie. Il existe plusieurs catégories de réseaux, chacune ayant des caractéristiques propres pour la topologie du réseau, les fonctions d'agrégation et de seuillage ou l'algorithme d'apprentissage. Pour plus de détails, nous renvoyons le lecteur à [DMS+02] pour une bonne introduction aux réseaux de neurones.

En analyse d'images, les réseaux de neurones peuvent être utilisés pour classer des pixels en fonction de leur couleur et éventuellement de leur voisinage.

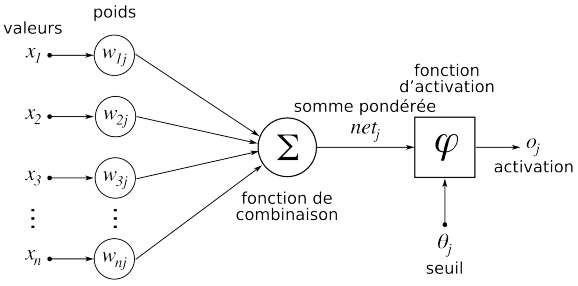


FIG. 2.8: *Structure d'un neurone artificiel. La valeur de sortie du neurone est le fruit de sa fonction d'activation appliquée sur la somme de ses entrées.*

Yan [Yan93] propose d'utiliser un réseau de neurones de type perceptron multi-couches [Ros58] pour classer d'une part la couleur de chaque pixel et d'autre part pour

distinguer les caractères des objets linéaires. Cette distinction se fait grâce à la particularité de cette technique qui est qu'elle ne s'intéresse pas simplement à la propriété colorimétrique du pixel, mais prend également en compte son voisinage. Cette approche semble robuste et efficace, mais elle nécessite une lourde phase d'apprentissage supervisé par l'opérateur.

Bessaid *et al.* [BBF03] propose une technique de segmentation de cartes basée sur les cartes auto-adaptatives (en anglais *self organizing map*) de Kohonen [Koh89]. Une carte de Kohonen constitue un réseau de neurones non supervisé où chaque neurone de la couche d'entrée est relié à chaque neurone de la couche de la carte. En fonction des valeurs d'entrée, un des neurones de la carte réagira mieux et sera gratifié de manière à ce qu'il réponde encore mieux à un autre stimulus de même nature. Dans le cadre de la segmentation d'images couleur proposée, les neurones d'entrées du réseau correspondent aux trois composantes RGB tandis que les neurones de sortie correspondent aux différentes classes de couleurs que l'on cherche à segmenter (par exemple marron, noir, bleu, vert et blanc). Avant d'être appliqué sur l'intégralité de la carte, le réseau est préalablement entraîné sur des échantillons de cartes sélectionnés par l'opérateur. La figure 2.9 illustre les résultats de la segmentation ainsi obtenue.

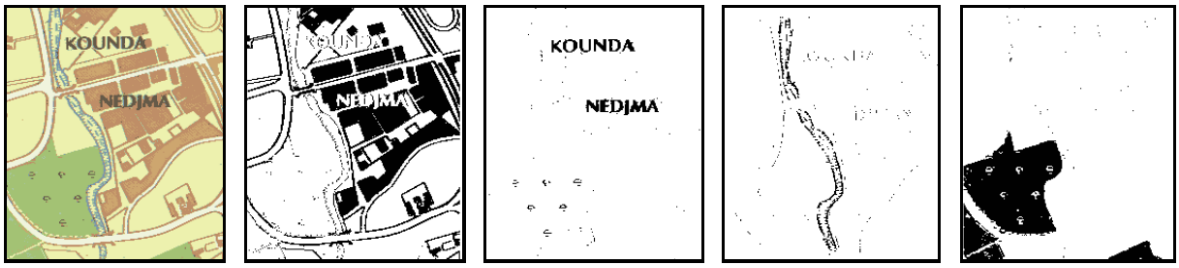


FIG. 2.9: *Segmentation par cartes auto-adaptatives de Kohonen obtenue par Bessaid [BBF03].*

2.4 Post-traitements

Une fois l'image segmentée et les courbes de niveau binarisées, on obtient une image où se distinguent les pixels objets, c'est-à-dire appartenant à une courbe de niveau, des pixels du fond. Il convient alors généralement d'appliquer différents algorithmes de traitement d'image tels que les outils de morphologie mathématiques.

Dans un premier temps, il faut analyser l'image pour supprimer des points isolés reconnus à tort comme faisant partie d'une courbe. Différentes techniques peuvent être utilisées. Le plus souvent, un filtrage des pixels isolés (par exemple par comptage du nombre de ses voisins activés) et des petites composantes connexes donne de bons résultats.

D'autre part, il arrive également que les courbes obtenues ne soient pas uniformes et que des pixels aient été considérés comme faisant partie du fond alors qu'ils se trouvaient en réalité sur une courbe. L'opération de fermeture morphologique (opération de dilatation puis d'érosion des objets) se prête bien à ce genre de corrections.

La figure 2.10 illustre le résultat d'une procédure de post-traitement constituée de la suppression des composantes connexes de taille inférieure à 10 pixels et d'une fermeture morphologique.

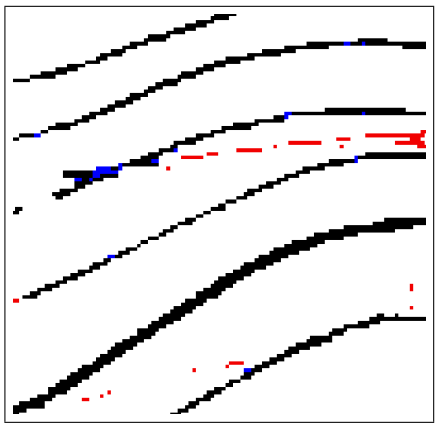


FIG. 2.10: *Exemple de post-traitements des courbes de niveau segmentées. Les pixels rouges correspondent aux pixels éliminés par la procédure, en bleu les pixels ayant été ajoutés.*

2.5 Bilan

La segmentation de cartes topographiques couleur n'est pas une chose aisée. En effet, les courbes de niveau sont des objets linéaires tracés avec une couleur (souvent marron / orange mais pouvant varier selon l'éditeur) pouvant être utilisée pour décrire d'autres objets et les techniques d'impression et de numérisation des cartes peuvent entraîner de réelles difficultés d'extraction. Si de nombreuses méthodes peuvent être utilisées pour classifier efficacement les différents éléments de la carte, et en particulier les courbes de niveau, il n'existe pas de technique donnant toujours de bons résultats quelle que soit la carte étudiée. Certaines méthodes requièrent un paramétrage manuel délicat pour déterminer la plage de couleur utilisée pour décrire les courbes, d'autres nécessitent un apprentissage à partir d'échantillons sélectionnés par l'opérateur. En fonction de la segmentation obtenue, différents post-traitements peuvent éventuellement être nécessaires pour obtenir une carte binaire distinguant au mieux les pixels appartenant à une courbe du reste. En définitive, si les traitements peuvent être en partie automatisés pour traiter plusieurs extraits d'une même carte ou plusieurs cartes du même type, l'expertise de l'opérateur reste toujours nécessaire déterminer la meilleure méthode à appliquer et guider l'analyse en corrigeant parfois manuellement les erreurs engendrées pouvant nuire au bon déroulement des étapes suivantes de reconstruction et éventuellement d'interpolation. L'utilisation d'un logiciel adapté est donc nécessaire pour effectuer ces traitements.

Chapitre 3

Reconstruction des courbes de niveau

3.1 Introduction

La phase de reconstruction (ou de reconnexion) des courbes de niveau consiste en réalité en une vectorisation. Il s'agit d'un traitement des courbes ou morceaux de courbes disponibles sous une forme raster afin d'en obtenir une version vectorielle. Dans un premier temps, les courbes de niveau sont squelettisées afin de leur donner une épaisseur de 1 pixel. Dans un deuxième temps, on cherche à reconnecter les morceaux de courbes de niveau qui se trouvent isolés soit à la suite d'une erreur dans la phase de segmentation (mauvais choix d'un seuil par exemple), soit du fait de leur non-connexité dans la carte originale à cause d'un objet superposé. Dans tous les cas, cette reconnexion est nécessaire pour faciliter la phase d'affectation des altitudes et pour ne pas perdre de la précision, voire entraîner des erreurs lors de l'interpolation du MNT.

Dans ce chapitre nous présentons d'abord (section 3.2) les différentes techniques pouvant être utilisées pour squeletisser les courbes de niveau. Dans la section 3.3, nous introduisons les méthodes existantes pour reconstruire les différents morceaux de courbes. Nous présentons ensuite, dans la section 3.4, une technique originale que nous avons mise au point et permettant de reconstruire sans paramètre et de façon naturelle les trous entre les morceaux de courbes à l'aide d'un champ d'orientation reconstruit à partir des courbes initiales. Enfin, la cotation des courbes, consistant à leur affecter l'altitude de la coupe de terrain représentée est discutée, dans la section 3.5.

3.2 Squelettisation des courbes de niveau

La squelettisation est un problème classique en analyse d'image. Elle consiste à réduire un objet discret en un ensemble de segments discrets d'épaisseur égale à 1 pixel appelé *squelette* ou *axe médian* et centré dans la forme d'origine. Il existe différentes familles d'algorithmes pour construire des squelettes :

- simulation de propagation de feu de prairie [Xia89] (voir figure 3.1) ;
- diagramme de Voronoï [Att95] ;
- extraction des maximas locaux de la carte de distance [AB89] [PDSS98] ;

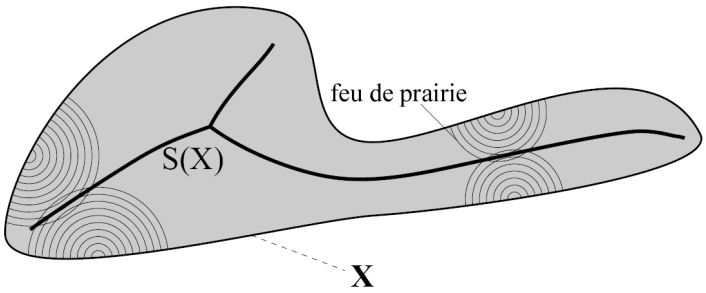


FIG. 3.1: *Le squelette peut être défini comme l'ensemble des pixels où les fronts des feux démarrant aux frontières de l'objet se rencontrent.*

– amincissements successifs par filtres de morphologie mathématique du type transformation de voisinage (*hit and miss transform*) [Ser82] [GW92].

Nous utilisons cette dernière pour sa facilité de mise en œuvre et ses bons résultats dans le cadre de notre étude. La squelettisation est réalisée par des transformations morphologiques simples. Elle est alors vue comme un processus d'érosion des objets qui en préserve la connexité.

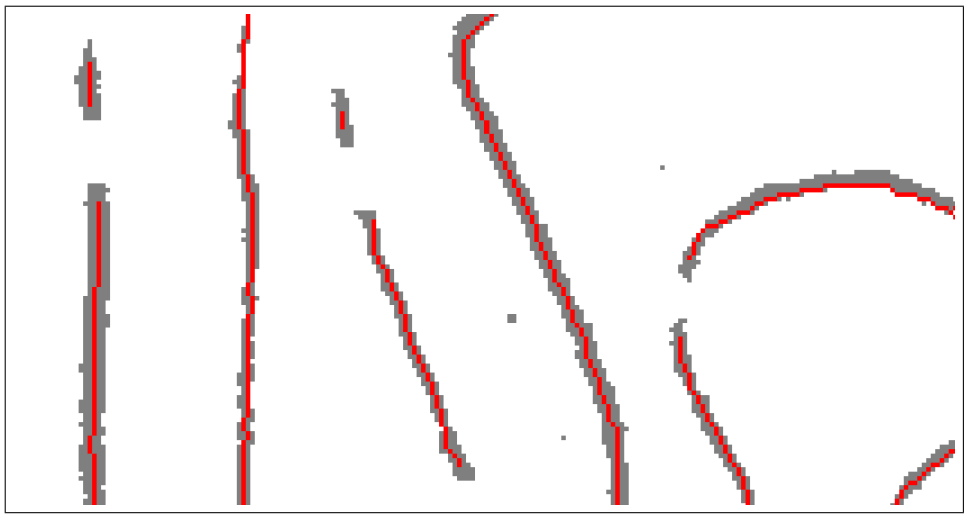


FIG. 3.2: *Courbes de niveau extraites (en gris) et leur squelette (en rouge) obtenu par amincissements successifs.*

La figure 3.2 montre la squelettisation de courbes de niveau par amincissement. On peut constater la perte d'informations au niveau de l'extrémité des courbes, le processus de squelettisation ayant comme effet de les rogner. Arrighi *et al.* [Arr98] proposent de détecter le point représentant l'extrémité de la courbe avant d'appliquer la méthode de squelettisation par points d'ancrage de Vincent [Vin91].

3.3 Techniques de reconstruction existantes

3.3.1 Approches basées image

Les approches les plus souvent proposées dans les solutions de vectorisation et de reconstruction de courbes de niveau sont basées sur des principes perceptifs simples. Par exemple, la décision de fermer ou grouper deux ensembles de segments ou pixels différents se base souvent sur les deux critères que sont la proximité et la continuité. La figure 3.3 illustre un exemple de résultat obtenu avec le logiciel commercial R2V. A l'évidence, cette reconstruction qui semble reposer uniquement sur la proximité n'est pas acceptable dans le cas qui nous intéresse.

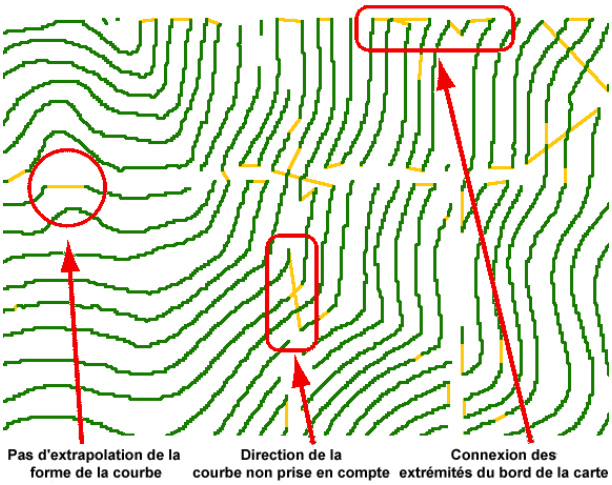


FIG. 3.3: *Exemple de reconstruction basée image obtenue à l'aide du logiciel commercial R2V. On constate 3 types d'erreurs : pas d'extrapolation lisse du segment reconstruit, pas de prise en compte de la direction de la courbe reconstruite et jointures mal gérées au niveau des bords de l'image* (source : Easy Trace Group.)

Dans [EAK95], Eikvil *et al.* une technique de conversion raster vers vecteurs est présentée en vue de traiter des cartes papiers numérisées. Le problème de la reconstruction est résolu en assumant qu'il n'y a qu'une continuation possible pour chaque extrémité. Ce prolongement naturel peut être trouvé en suivant la direction de la ligne à l'extrémité. Arrighi *et al.* [AS99], utilisent une combinaison entre la distance euclidienne entre les extrémités et les différences entre les directions des tangentes en ces points.

Malgré son attractivité due à sa simplicité, tous les algorithmes existants de reconstruction basés sur des critères locaux échouent le plus souvent sur des cas simples (voir figure 3.4).

3.3.2 Approches géométriques

La reconstruction de courbes peut être analysée comme un cas d'un problème de reconstruction géométrique plus général. Soit un ensemble fini d'échantillons V d'une courbe λ, la tâche de reconstruction consiste à construire un graphe $G = (V, E)$ de façon à ce que deux points de V soient reliés par une arête si et seulement si les points sont adjacents dans λ. Le graphe G est appelé reconstruction polygonale de λ. Le problème de reconstruction de courbes a été sujet à de nombreux travaux dans la communauté de géométrie computationnelle.

Amenta *et al.* [ABE98] introduisent le concept de largeur locale (*local feature size*) correspondant à la distance d'un point à l'axe médian de sa courbe. En utilisant ce concept, ils définissent une condition d'échantillonnage non uniforme permettant une reconstruction

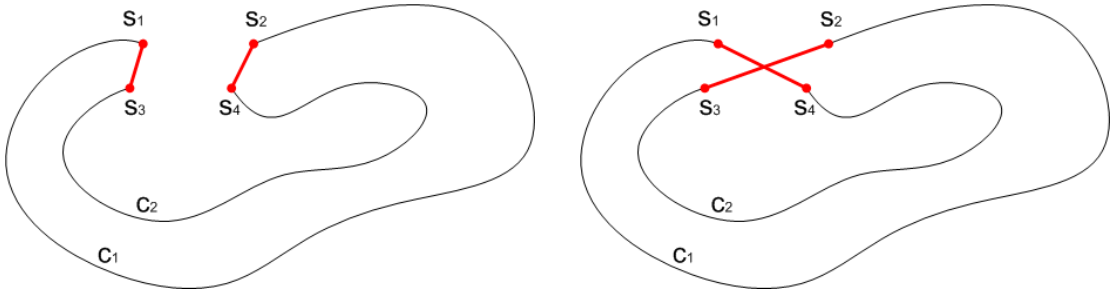


FIG. 3.4: *Problèmes de la reconstruction basée image. A gauche : reconstruction sur le critère de distance, à droite sur le critère d'orientation de la tangente à l'extrémité de la courbe.*

à partir d'échantillons de densités diverses. L'algorithme proposé calcule la triangulation de Delaunay de l'ensemble des points puis la filtre afin d'obtenir la reconstruction. Dey *et al.* [DRM99] ont étendu cette technique pour prendre en compte un ensemble de courbes lisses ouvertes ou fermées en utilisant un filtrage de la triangulation de Delaunay. Dans [DW00], les auteurs présentent un algorithme qui prend correctement en compte les coins et les points terminaux. Cependant, l'algorithme souffre de défauts et il est facile de trouver des exemples où il échoue. Ces approches donnent de bons résultats dans l'ensemble, mais reposent uniquement sur des critères géométriques. Plus récemment, Spinello *et al.* [SG04] proposent de vectoriser les courbes de niveau en utilisant la triangulation de Delaunay puis de filtrer les arêtes sur des critères locaux et globaux. Cependant, l'approche globale utilisée n'est pas optimale car elle est fondée sur un algorithme glouton et les règles de connexion sont une fois encore purement géométriques.

Dans [DDG98a], Dupont *et al.* utilisent un système expert basé sur l'analyse locale de la géométrie, les relations de voisinage et le diagramme de Voronoï.

3.3.3 Approches par champ de vecteurs gradients

Notre méthode repose sur un champ d'orientation obtenu à l'aide des courbes de niveau disponibles en entrée. Le champ d'orientation a déjà été utilisé en analyse d'images ; par exemple [CFCK06], montre comment un opérateur d'interpolation de l'orientation peut être utilisé pour retrouver des informations géométriques dans des images. Une autre application aux modèles déformables de type *snakes* est donnée dans [GR05].

3.4 Reconstruction globale à l'aide d'un champ d'orientation

Dans cette section, nous nous intéressons au problème spécifique de reconstruction des courbes de niveau. Nous supposons que nous disposons donc d'une carte binaire sur lesquels les courbes ont été préalablement extraites puis affinées.

L'idée de base de cette technique, présentée à ICDAR 2007 [PS07], est de reconstruire le champ d'orientation du gradient des courbes de niveau disponibles et de l'utiliser d'une façon globale pour déterminer les paires naturelles d'extrémités qui devraient être connectées. Une fois que l'appariement des extrémités a été obtenu, nous utilisons le champ d'orientation pour assurer une reconstruction lisse.

Notre algorithme peut être résumé en trois grandes étapes qui sont :

1. la création du champ d'orientation à partir des normales aux courbes d'origine ;
2. l'appariement des extrémités ;
3. le remplissage des trous en chaque paire.

3.4.1 Création du champ d'orientation

La première étape de l'algorithme consiste à créer le champ d'orientation sur toute la carte à partir de l'orientation des normales calculées sur les courbes de niveau disponibles en entrée. Il est important de noter que nous nous intéressons uniquement à la direction des vecteurs normaux et non à leur sens. Nous noterons F le champ d'orientation de dimension $m \times n$ (c'est à dire la taille de l'image de la carte). Les valeurs prises dans ce champ sont des valeurs d'angles réels compris entre $]-\pi, \pi[$.

C'est sur ce champ d'orientation que repose notre méthode, aussi nous mettons une attention toute particulière pour le calculer. Ce calcul est réalisé en deux étapes : i) calcul de l'orientation des normales pour chaque pixel des courbes de niveau disponibles ; ii) interpolation des orientations disponibles sur l'ensemble des pixels du fond.

Calcul des normales aux contours Le calcul des vecteurs normaux pour chaque pixel d'une courbe discrète 2D n'est pas un problème trivial. En géométrie discrète, des méthodes spécifiques d'estimation de tangentes de courbes ont été étudiées. Nous renvoyons le lecteur à [LVdV05] pour une étude des techniques les plus efficaces. Cependant, ces techniques ne garantissent pas la continuité des valeurs de tangentes le long de la courbe. Après quelques tests, nous nous sommes rendu compte que cette contrainte ne permettait pas d'obtenir un champ d'orientation suffisamment continu. Notre choix s'est alors porté sur une approche par B-spline : chaque courbe de niveau est interpolée par une B-spline et la tangente (et donc la normale) est ensuite calculée avec précision.

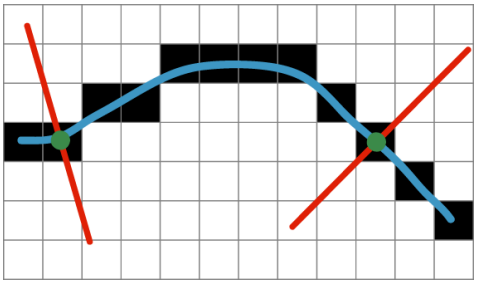


FIG. 3.5: *Orientation des normales sur une B-spline interpolant une courbe de niveau discrète.*

L'interpolation des courbes de niveau par des B-splines peut être vue comme une opération de conversion raster vers vecteurs. Cette opération est rendue aisée par l'opération d'affinage des courbes en composantes 8-connexes. Le calcul des vecteurs normaux des B-splines est effectué par évaluation à l'aide de l'algorithme de De Boor (voir figure 3.5). L'angle du vecteur normal est alors stocké dans le champ d'orientation F.

Interpolation du champ Une fois que l'orientation des normales est connue sur les courbes de niveau, nous procédons à leur interpolation sur l'ensemble du champ F et donc en chaque point de l'image. Un tel problème d'interpolation est traditionnellement résolu en résolvant

Input : Le champ d'orientation F initialisé sur les courbes
Result : Le champ interpolé F
File Q_1, Q_2
for *chaque pixel* p *de* F **do**
  if $F(p) \neq \infty$ **then**
    $Q_1.push(p)$
  end
end
$s \leftarrow 0$
while $Q_1.size \neq s$ **do**
  $s \leftarrow Q_1.size$
  while $Q_1.size \neq 0$ **do**
    $p \leftarrow Q_1.pop()$
    $\theta \leftarrow \Theta(p, F)$
    if p *n'est pas sur une courbe* **then**
      $F(p) \leftarrow \theta$
    end
    $Q_2.push(p)$
    foreach $q \in N(p)$ **do**
      if $F(q) = \infty$ **then**
        $F(q) \leftarrow \theta$
        $Q_2.push(q)$
      end
    end
  end
  échange(Q_1, Q_2)
end

Algorithme 1 : Fonction `InterpolationOrientation()`

une équation aux dérivées partielles (comme le Laplacien). Par exemple, un opérateur d'interpolation AMLE est présenté dans Chessel *et al.* [CFCK06]. Cependant, nous proposons une implémentation rapide basée sur une propagation des valeurs connues. Celle-ci est décrite dans l'algorithme 1. La fonction Θ est une fonction clé qui calcule l'orientation moyenne en un point p en utilisant une fenêtre 3×3 dans le 8-voisinage $N(p)$ de p :

$$\Theta(p) = \frac{\sum_{q \in N(p)} \Lambda(F(q), F(p))}{|N(p)|}. \tag{3.1}$$

La fonction Λ assure la consistance des opérations sur les orientations entre deux angles α et *beta* du champ F.Elle est définie comme suit :

$$\Lambda(\alpha, \beta) = \begin{cases} \alpha & if d1 = min(d1, d2, d3) \\ \alpha + \pi & if d2 = min(d1, d2, d3) \\ \alpha - \pi & if d3 = min(d1, d2, d3) \end{cases} \tag{3.2}$$

avec $d1 = |\beta - \alpha|$, $d2 = |\beta - \alpha - \pi|$ et $d3 = |\beta - \alpha + \pi|$.

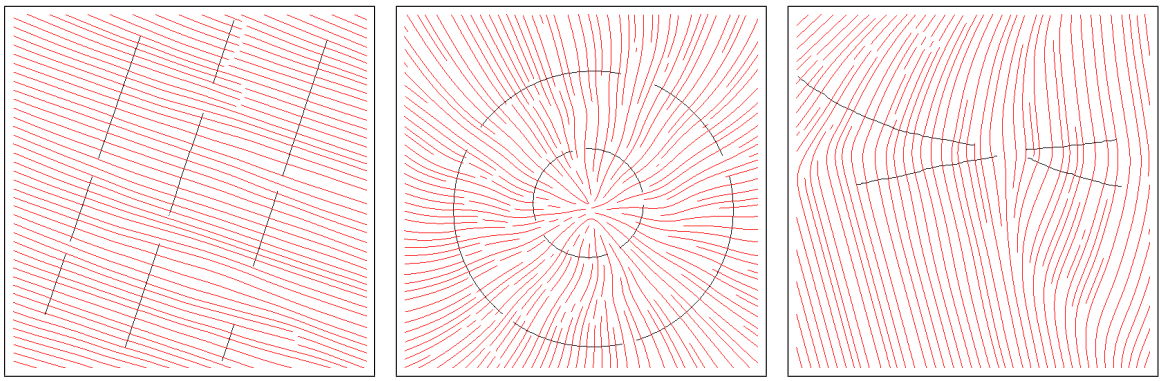


FIG. 3.6: *Lignes de flux des champs d'orientations interpolés à partir des normales calculées sur des échantillons de courbes altérées.*

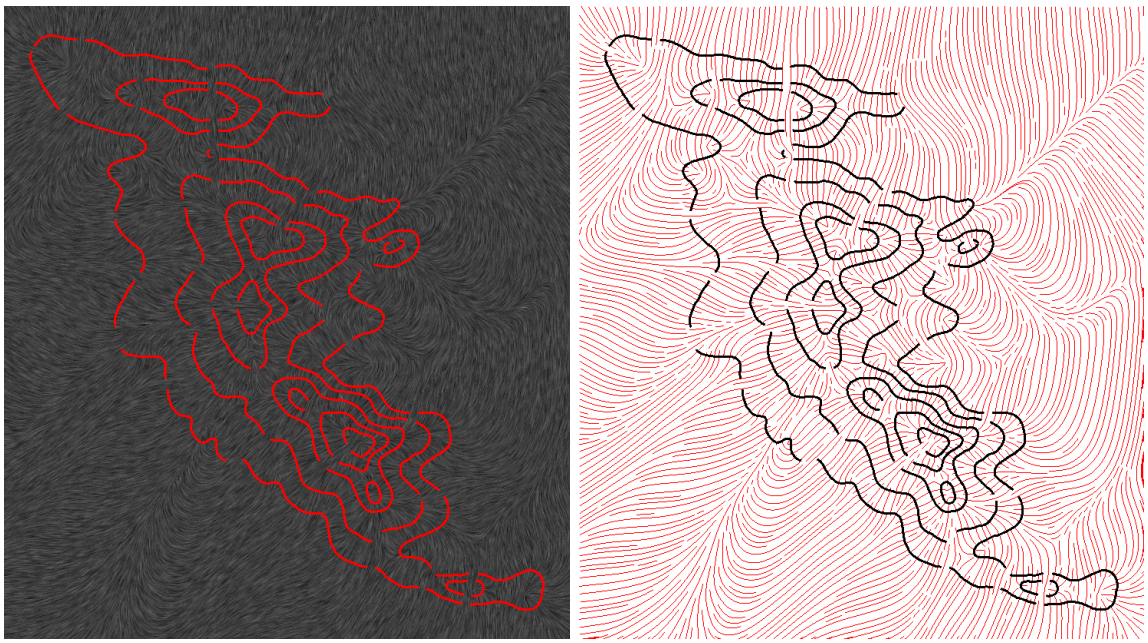


FIG. 3.7: *LIC et lignes de flux du champ d'orientation interpolé à partir des normales calculées sur des courbes de niveau altérées.*

La figure 3.6 montre les lignes de flux du champ d'orientation obtenu à l'aide de l'algorithme de visualisation de streamlines décrit par Jobard *et al.* [JL97] que nous avons adapté aux champs d'orientation. Un autre exemple est donné dans la figure 3.7 avec une visualisation du champ d'orientation à l'aide de la méthode LIC (*line integral convolution*) proposée par Cabral *et al.* [CL93].

3.4.2 Connexion des extrémités

La reconstruction des extrémités se fait par appariement des extrémités. Un tel problème est bien connu en théorie des graphes en tant que problème d'appariement. Dans cette partie, nous cherchons à déterminer quelles paires d'extrémités devraient être connectées ensemble. Ceci est réalisé en deux étapes :

1. calculer l'énergie nécessaire pour aller d'une extrémité à l'autre à travers le champ d'orientation ;
2. résoudre le problème d'appariement parfait.

Estimation du poids d'appariement Pour chaque paire d'extrémités, nous associons un poids résultat d'une fonction d'énergie. Cette fonction doit être définie de façon à ce que : i) l'énergie soit nulle si les extrémités se rencontrent directement en suivant la courbe de niveau du champ ; ii) l'énergie soit maximale proportionnellement à la longueur du chemin quand les deux extrémités se rejoignent en suivant la ligne de champ du champ d'orientation. Nous proposons une fonction d'énergie calculée comme suit :

$$\omega(e_1, e_2) = \sum_{t=0}^{1} \left| \overrightarrow{p - q} \cdot \overrightarrow{F(p)} \right| . \tag{3.3}$$

La figure 3.11 illustre le schéma de reconstruction : les points des courbes de niveau originales C_1 et C_2 sont représentées en noir, en gris sont dessinées les courbes de niveau du champ F issues de chacune des extrémités e_1 and e_2, enfin, les pixels bleus représentent le morceau de courbe reconstruit en utilisant l'algorithme décrit plus loin (section 3.4.3).

Appariement parfait Une fois la matrice W calculée, nous procédons à l'appariement parfait des extrémités. Soit $G(V, E)$ un graphe et $w : E \to \mathbb{R}$ une fonction de coût, dans notre cas la fonction 3.3. Un *appariement parfait* de G est un sous-ensemble $M \subset E$ tel que chaque nœud de G est incident à exactement une arête de M. On note alors $w(M)$ le poids de l'appariement M est la somme des poids de ses arêtes. Le problème consiste donc à trouver l'appariement parfait de poids minimal (ou maximal, voir l'exemple figure 3.8). Ce problème peut être résolu en temps polynomial en utilisant l'algorithme d'Edmonds [Edm65]. Nous utilisons une implémentation en $O(N^3)$ basée sur les travaux de recherche de Gabow [Gab74].

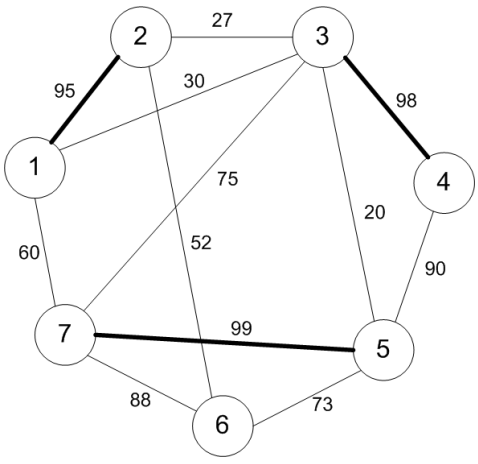


FIG. 3.8: *Appariement parfait de coût maximum d'un graphe à 7 sommets.*

Cas des extrémités proches du bord Avant d'effectuer l'appariement, une première étape permet d'éliminer les extrémités proches du bord et dont la courbe de niveau issue

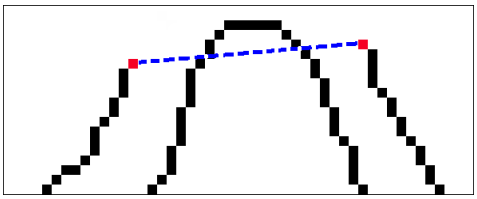


FIG. 3.9: *Problème de la reconstruction basée image.*

du champ d'orientation atteint le bord de l'image sur une distance inférieure à τ pixels (expérimentalement, la valeur $\tau = 200$ est satisfaisante). Le morceau de courbe reconstruit pour atteindre le bord est simplement constitué des pixels traversés par la courbe de niveau du champ d'orientation (voir figure 3.10).

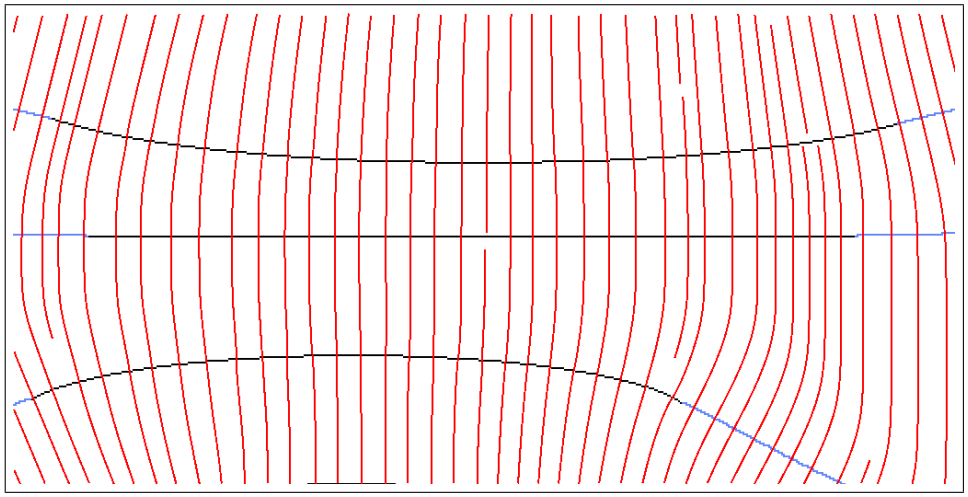


FIG. 3.10: *Reconstruction des courbes proches des bords à l'aide du champ d'orientation.*

3.4.3 Reconstruction lisse

Une fois l'appariement obtenu, il reste à reconstruire les morceaux de courbes manquants entre chaque paire d'extrémités. Différentes méthodes peuvent être utilisées : la plus simple consiste à tracer un segment de ligne. Cependant, pour des raisons évidentes, dans la plupart des cas, le résultat obtenu n'est pas satisfaisant, car elle ne reflète pas la courbure naturelle de la courbe. De plus, des erreurs d'ordres topologiques peuvent survenir, par exemple en croisant une courbe préexistante, comme illustrée dans la figure 3.9. Un meilleur résultat visuel peut être obtenu en traçant une courbe de Bezier ou en faisant une interpolation de Hermite à l'aide des tangentes $\overrightarrow{t_1}$ et $\overrightarrow{t_2}$ calculées aux extrémités. Cependant, si la courbe est visuellement plus naturelle, rien ne garantit l'absence de création de problèmes topologiques. Pour remédier à cela, nous proposons d'utiliser un processus d'interpolation pour construire un morceau de courbe lisse et naturel en s'appuyant sur le champ d'orientation disponible, comme l'illustre la figure 3.11.

Pour chaque extrémité, nous stockons les points de la courbe de niveau (dans ce cas, les points obtenus par intégration du gradient du champ d'orientation), c'est-à-dire les points

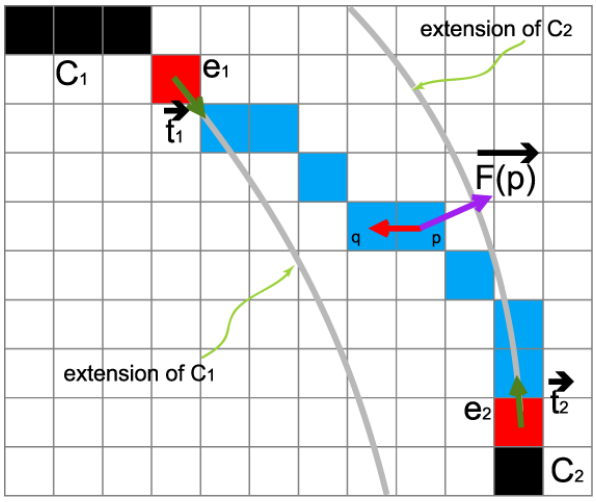


FIG. 3.11: *Courbe reconstruite entre une paire d'extrémités à l'aide du champ d'orientation.*

qui prolongent naturellement la courbe de niveau dans le champ d'orientation reconstruit (en gris sur la figure 3.11).

Comme nous l'avons dit, l'idée de base consiste à interpoler linéairement les courbes issues du champ à partir des extrémités e_1 et e_2. Les points sont stockés dans 2 files Q_{e_1} et Q_{e_2} de tailles respectives $S(Q_{e_1})$ et $S(Q_{e_2})$. Les coordonnées de ces points sont paramétrées comme suit :

$$p(t) = \begin{pmatrix} x_{p_{e_1}(t)} * (1-t) + x_{p_{e_2}(t)} * t \\ y_{p_{e_1}(t)} * (1-t) + y_{p_{e_2}(t)} * t \end{pmatrix} \tag{3.4}$$

avec t prenant ses valeurs dans l'intervalle $[0, 1]$,

$$\begin{aligned} p_{e_1}(t) &= Q_{e_1}[S(Q_{e_1}) * t/d], \\ p_{e_2}(t) &= Q_{e_2}[S(Q_{e_2}) - (S(Q_2) * t/d)]. \end{aligned}$$

$Q_{e_1}[x]$ correspond au x-ième point la la file Q_{e_1} et d est la distance Euclidienne entre e_1 et e_2.

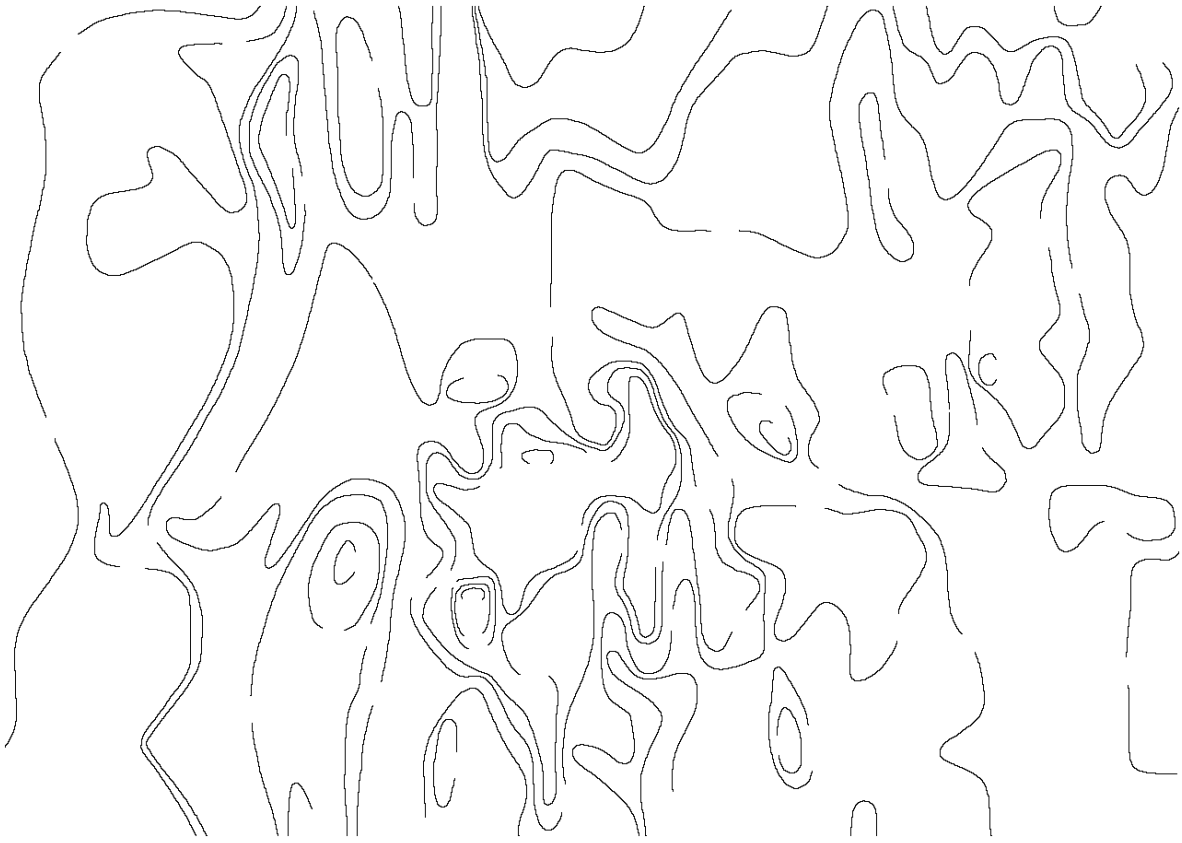
3.4.4 Résultats

Les figures 3.12 et 3.13 illustrent les différentes étapes de notre technique. La taille de la carte est 1278×903 pixels. Sur un processeur Intel Core 2 Duo cadencé à 1.86GHz, le temps de calcul total est de 23 secondes : 22 secondes pour la construction et l'interpolation du champ d'orientation et moins d'1 seconde pour le calcul de l'appariement et la reconstruction. Nous constatons que 100% des courbes ont été correctement restaurées et que les parties reconstruites sont lisses et semblent naturelles.

La figure 3.14 montre le résultat de la reconstruction obtenue sur une autre carte topographique.

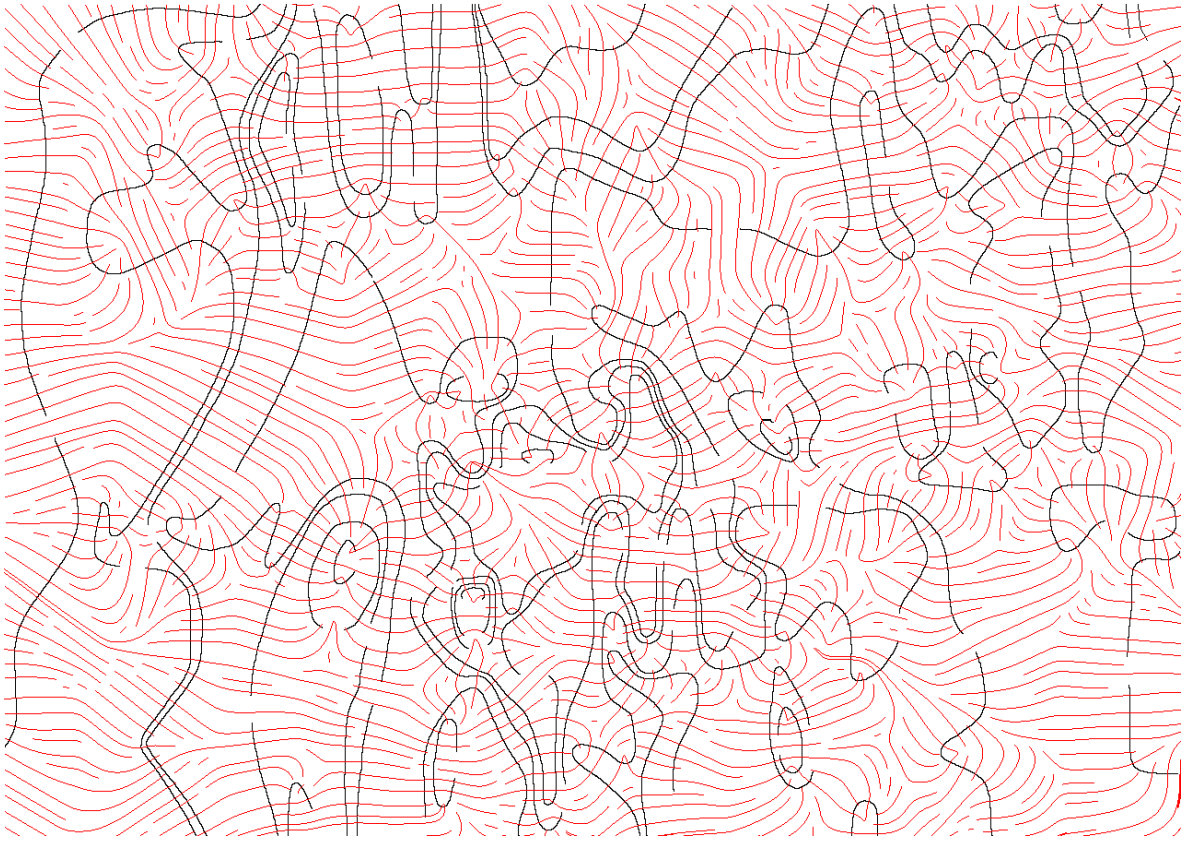
3.5 Cotation des courbes de niveau

(a) Carte bathymétrique numérisée.

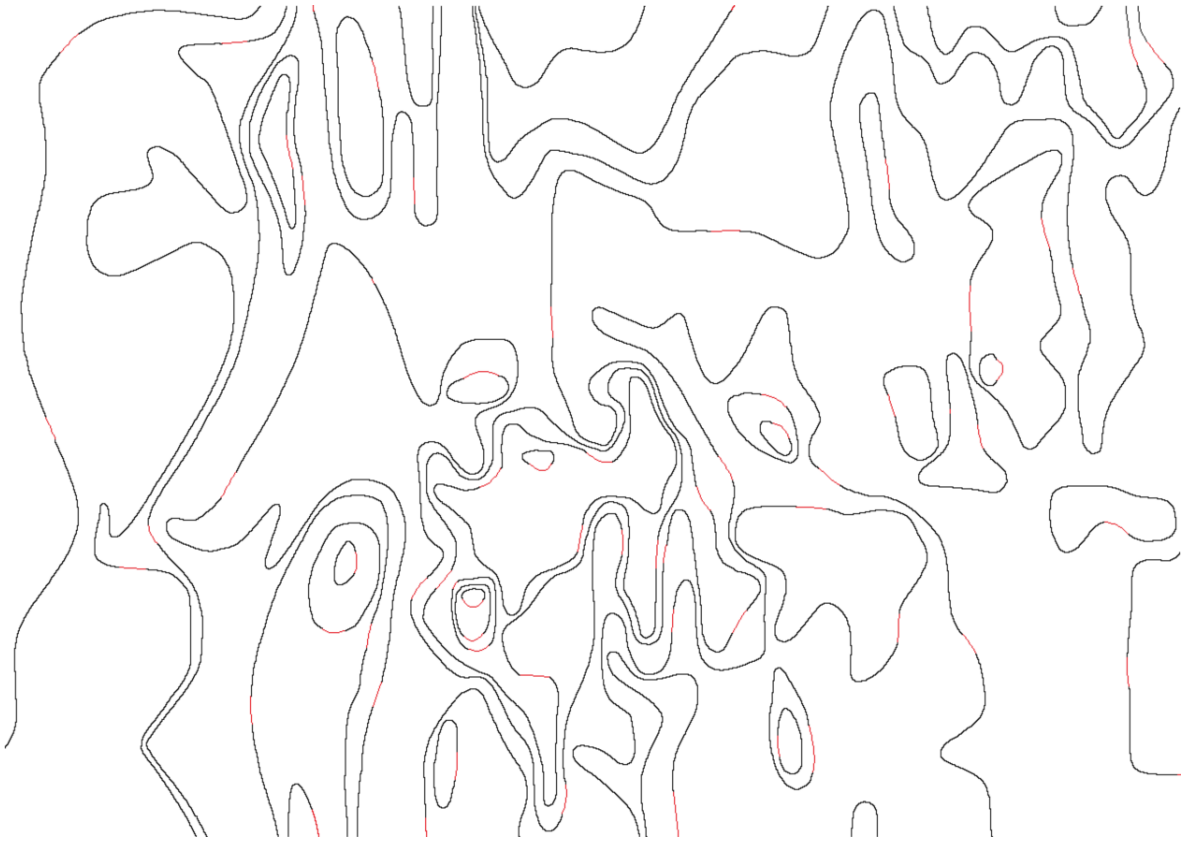


(b) Courbes de niveau vectorisées sous forme de B-Splines.

FIG. 3.12: *Reconstruction des courbes de niveau sur une carte bathymétrique du lac Winnibigoshish dans Minnesota, Etats-Unis.*



(a) Lignes de flux du champ d'orientation interpolé à partir des normales calculées sur les courbes de niveau.



(b) Reconstruction obtenue. Les morceaux de courbes reconstruits sont en rouge.

FIG. 3.13: *Reconstruction des courbes de niveau sur une carte bathymétrique du lac Winnibigoshish dans Minnesota, Etats-Unis.*

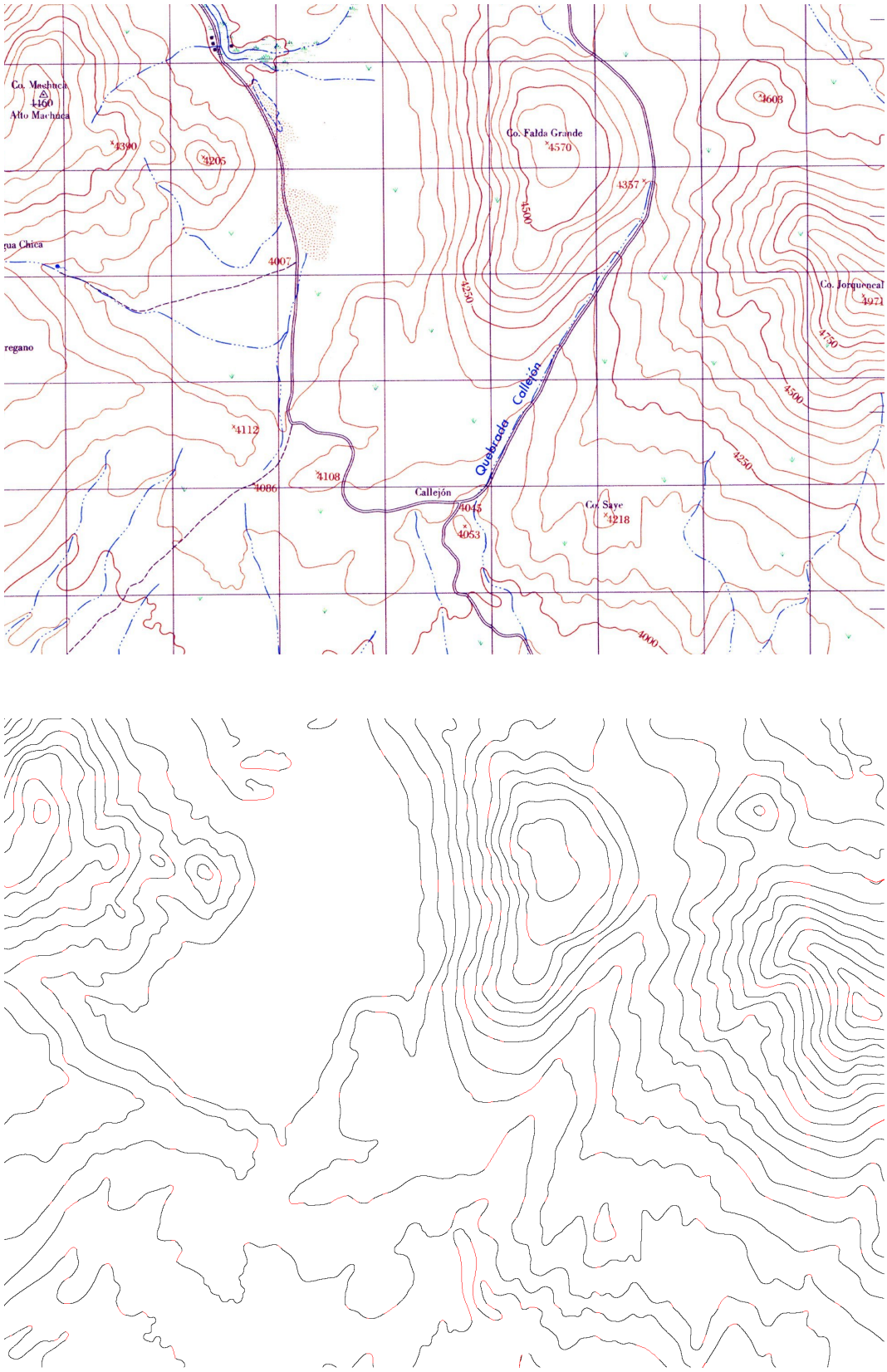


FIG. 3.14: *Renconstruction des courbes de niveau d'un extrait de carte topographique après segmentation. Les morceaux de courbes reconstruits sont en rouge dans l'image du bas.*

Une fois les courbes reconstruites, l'étape suivante consiste à les coter avec leur altitude. Durant cette thèse, nous n'avons pas mené d'investigation particulière sur cette étape et proposons simplement, dans notre logiciel `AutoDEM`, des outils d'affectation nécessitant des interventions de l'opérateur (voir figure 3.15.

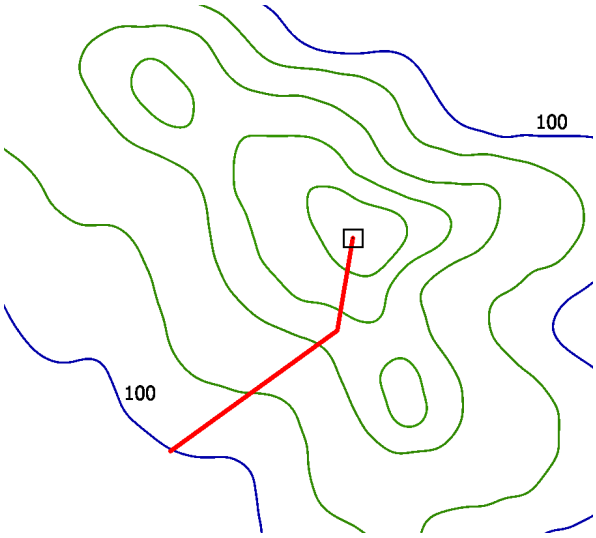


FIG. 3.15: *Quotation manuelle des courbes de niveau à l'aide d'un outil de propagation à partir d'une courbe cotée le long d'une polyligne.*

Le problème principal d'une approche automatique est d'effectuer la quotation des autres courbes à partir de la cotation de quelques courbes particulières. Dans la mesure où le pas entre chaque courbe varie entre plus ou moins un certain pas fixé pour une carte donnée, l'opération peut sembler aisée. En réalité de nombreux cas ambigus peuvent survenir si les courbes initialement cotées ne sont pas bien choisies.

Il existe toutefois différents travaux pour tenter d'automatiser le plus possible cette étape. Dans [DDG98a, DDG99, Dup99], Dupont *et al.* utilisent des informations supplémentaires permettant d'estimer le sens de pente soit par une technique de reconnaissance de caractères [PD94] pour déterminer l'altitude des courbes de niveau maîtresses plus épaisses, soit en utilisant un MNT préexistant, le plus souvent de moindre précision. Les pentes estimées permettent ensuite de maintenir une cohérence globale du résultat.

Si cette technique donne de bons résultats dans le cas où une telle information supplémentaire est disponible (c'est désormais le cas pour des cartes à de grandes échelles représentant la topographie actuelle d'une région où l'on peut utiliser par exemple le MNA de la mission SRTM), elle ne permet pas de répondre aux cas des cartes décrivant des régions anciennes ou très localisées. Pour effectuer une cotation plus automatique que manuelle de ces cartes, nous pensons que l'utilisation du champ d'orientation décrit dans la section précédente pourrait être une base intéressante dans de futurs travaux.

3.6 Conclusion et travaux futurs

La reconstruction de courbes de niveau repose sur une étape vectorisation avancée dans le but de reconstituer des courbes fermées afin de simplifier la phase de cotation et la suite

des traitements. Une fois les composantes connexes des morceaux de courbes squelettisées, la principale difficulté réside dans la reconnexion des différents morceaux entre eux.

Dans ce chapitre, nous avons introduit une technique simple à utiliser (sans paramètre) et efficace pour reconstruire les courbes de niveau. Notre méthode est basée sur le champ d'orientation des normales extraites des courbes de niveau disponibles en entrée. L'aspect global de la reconstruction est basé à la fois sur l'utilisation de ce champ d'orientation global et de l'appariement parfait réalisé sur l'ensemble des extrémités potentiellement voisines. Les résultats obtenus sur différentes cartes sont très satisfaisants à la fois en terme de taux d'erreur et en termes topologique et visuel.

Nous pensons que la méthode de reconstruction du champ d'orientation décrit dans cette section pourrait être à la base d'une méthode de cotation semi-automatique.
Enfin, cette méthode a été appliquée pour reconstruire des courbes de niveau, mais elle pourrait tout aussi bien être utilisée avec succès dans d'autres problématiques d'analyse de documents et de vision par ordinateur.

Chapitre 4

Interpolation de MNT

4.1 Introduction

L'interpolation d'un MNT consiste à évaluer l'altitude des points le constituant à partir de points ou de courbes de niveau cotées. Cette construction du MNT se fait généralement en reconstruisant une fonction d'interpolation à partir des contraintes initiales (courbes de niveau ou points cotés) et en l'évaluant sur les points d'altitude inconnue. La littérature compte de nombreux travaux de recherche décrivant des techniques pour générer MNA ou RTI à partir de courbes de niveau et/ou points cotés. Dans cette thématique, notre travail s'est principalement porté sur le recensement et l'implémentation des différentes méthodes existantes pour en évaluer les propriétés. Nous avons également proposé une technique efficace pour interpoler de vastes MNT à partir d'échantillons épars.

Dans la section 4.2, nous présentons les techniques les plus couramment utilisées dans le cadre de l'interpolation à partir de courbes de niveau cotées ou de points cotés (valeurs d'altitudes non uniformément échantillonnées sur une surface 2D). Nous distinguons les techniques générant un MNA des techniques de triangulation générant un RTI. Nous verrons que ces dernières ne font pas forcément appel à un technique d'interpolation. Dans le cas de l'interpolation de MNA, nous supposons disposer en entrée d'un MNA initialisé avec des valeurs d'altitudes pouvant être le fruit d'une rastérisation de courbes de niveau ou pouvant avoir été obtenu par échantillonnage. Les points du MNA à déterminer sont eux initialisés avec une valeur d'altitude particulière (en pratique nous utilisons la valeur -32767).

Dans la section 4.3, nous présentons une technique efficace que nous avons mise au point pour interpoler un MNA lisse à partir d'un grand ensemble d'échantillons basée sur une utilisation hiérarchique en *quad-tree* des fonctions à base radiale.

Enfin, dans la section 4.4 nous comparons les différentes méthodes présentées selon différents critères (visuels, temps de calcul, etc.).

4.2 Les techniques d'interpolation

Notations

Dans la suite de cette section, nous utilisons les notations suivantes : nous notons $z : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}$ la fonction qui associe l'altitude à un point de la surface. Le point "générique" p peut aussi être notifié par ses coordonnées (x, y). z_p est le scalaire représentant l'altitude du point p, $z(p) = z_p$.

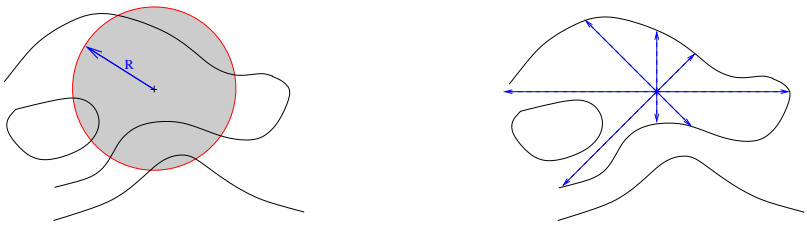
4.2.1 Inverse pondéré de la distance

La technique de l'inverse pondéré de la distance (en anglais *Inverse Distance Weighted Averaging*, IDWA) introduit une méthode simple pour interpoler une fonction 2D à partir d'un ensemble de points irrégulièrement espacés. Cette méthode se base sur une combinaison linéaire des valeurs des points connus en utilisant le principe que l'influence relative d'un point diminue avec la distance de l'endroit où l'on effectue l'observation de la même variable [Wat92]. De nombreuses variantes de cette technique ont été proposées. En 1968, Shepard [She68] propose d'utiliser les distances inverses pondérées par moindre carré. L'équation générale s'écrit sous la forme :

$$z(p) = \frac{\sum_{i=1}^{N(v_p)} d(p, p_i)^{-w} z_{p_i}}{\sum_{i=1}^{N(v_p)} d(p, p_i)^{-w}} \tag{4.1}$$

avec $d(p, p_i)$ la distance euclidienne entre p et p_i, w la puissance du poids (généralement $w \in \{1; 2\}$) et $N(v_p)$ le nombre de points connus considérés dans la voisinage v_p de p.

Un schéma global prenant en compte l'ensemble des points connus entraîne bien entendu un coût de calcul qui le rend rapidement inutilisable en pratique. On considère alors un ensemble de points localisés dans un disque de rayon fixé (voir figure 4.1(a)). Cette méthode fonctionne aussi bien avec des points cotés que des courbes de niveau. Pour être moins dépendant d'un paramètre de rayon, [JHJ86] et [Car88] proposent de détecter les points les plus proches pouvant être détectés en lançant des rayons dans différentes directions (entre 4 à 64, voir 4.1(b)) autour du point d'intérêt.



(a) Recherche dans un voisinage dans de rayon R.

(b) Recherche dans 8 directions.

FIG. 4.1: *Exemple de voisinages utilisés pour une interpolation IDWA.*

Pour l'interpolation de courbes de niveau, la première méthode ne donne pas de très bons résultats de par la difficulté de fixer le paramètre de rayon. Si le rayon est trop petit, des zones de plateaux sont générées au lieu d'une pente, si le rayon est trop large, la complexité de l'algorithme explose. La seconde méthode est plus précise si un grand nombre de rayons est envoyé. Cependant, dans de nombreux cas, des zones plates peuvent être générées : par exemple dans une courbe n'en englobant pas une autre.

Bien que cette méthode soit rapide et simple à implémenter, la surface générée est de type C^0 et rien n'assure que la pente soit correcte. Cependant, cette méthode est pratique

pour estimer en temps réel l'altitude d'un point sélectionné par le pointeur souris dans une application SIG.

Dans un papier récent, Gousie *et al.* [GF03] proposent une méthode hybride basée sur la génération de courbes de niveau intermédiaires. Les trous restants sont remplis à l'aide de la technique IDWA tandis que les pics et les puits sont interpolés à l'aide d'une spline de Hermite qui suit la direction de la pente. Le MNA est finalement obtenu après plusieurs phases de lissage, mais selon les auteurs, des artefacts persistent et la surface à tendance à être stratifiée.

4.2.2 Interpolation par Voisins Naturels

La méthode d'interpolation par voisins naturels fut introduite en 1980 par Sibson [Sib80]. Une présentation plus précise en est faite dans [Sib81]. Il s'agit d'une méthode populaire et utilisée dans de nombreux domaines. L'interpolation par voisins naturels est construite sur la base du diagramme de Voronoï calculé à partir des échantillons de données connus.

Cette méthode consiste en une moyenne pondérée par la distance aux voisins pour calculer la fonction d'interpolation. La différence fondamentale entre cette méthode et celle proposée par Shepard tient aux poids affectés aux voisins : on ne considère plus la distance mais l'aire que recouvre l'intersection du diagramme de Voronoï initial et le diagramme de Voronoï initial auquel on insère le point à interpoler (cf. figure 4.2).

A partir d'un ensemble de sites, on calcule d'abord le diagramme de Voronoï de ces sites (voir figure 4.2a). Pour interpoler la valeur au point p, p est inséré dans le diagramme de Voronoï (voir figure 4.2b). La cellule de Voronoï $V(p)$ de p a k cellules voisines appelées $V_1(p); \ldots ; V_k(p)$. Les k sites $p_1; \ldots ; p_k$ sont appelés les *voisins naturels* de p. L'aire de $V(p)$ est l'union des aires $\lambda_i(p)$ appartenant aux cellules $V_i(p)$ des voisins de p dans le diagramme initial. L'aire $\lambda_i(p)$ étant définie ainsi :

$$\lambda_i(p) = \frac{Aire(V(p) \cap V_i(p))}{Aire(V(p))}. \tag{4.2}$$

La fonction d'interpolation f est alors définie comme suit :

$$f(p) = \frac{\sum_{i=1}^{k} \lambda_i(p) f(p_i)}{\sum_{i=1}^{k} \lambda_i(p)}. \tag{4.3}$$

Un exemple de reconstruction avec cette fonction f est donné dans la figure 4.2. L'interpolation de Sibson est un schéma local dans la mesure où seules les valeurs des voisins naturels d'un point p influencent la valeur interpolée $f(p)$.

Cette méthode est la plus appropriée lorsque la densité des points échantillons est irrégulière.

4.2.3 Interpolation géodésique

Introduite par Soille [Soi91], l'interpolation géodésique est une technique d'interpolation adaptée à l'interpolation des courbes de niveau et non à l'interpolation d'échantillons épars. Le principe de l'interpolation géodésique est que chaque point du MNA est influencé par les deux courbes de niveau les plus proches. L'altitude d'un point donné s'obtient par combinaison linéaire des altitudes des deux courbes de niveau et de la distance géodésique entre le point et celles-ci.

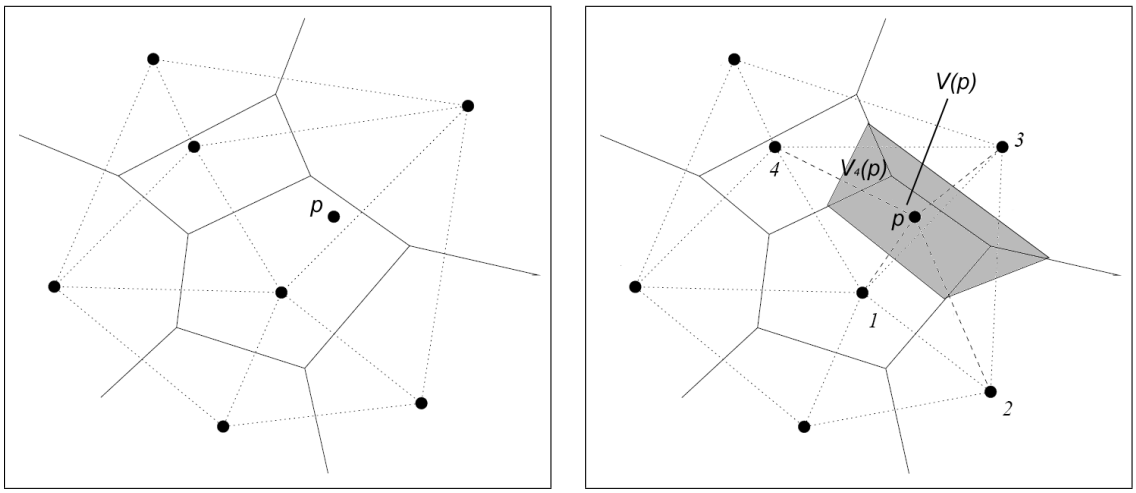


FIG. 4.2: *Interpolation par voisins naturels à partir de 7 sites. A gauche : diagramme de Voronoï des 7 sites initiaux. A droite : en gris la cellule de Voronoï du point à interpoler.*

La distance géodésique $d_A(p,q)$ entre deux points p et q est la longueur du plus court chemin les joignant inclus dans un ensemble connexe A :

$$d_A(p,q) = \inf l(P) \mid p_0 = p, p_l = q, and P \subset A. \tag{4.4}$$

Cette notion de distance entre deux points peut être étendue [LM84] à une distance géodésique entre un point p et un sous-ensemble Y de A et correspond à la plus petite distance géodésique entre p et tout point y de Y :

$$d_A(p,Y) = \inf_{y \in Y} d_A(p,y). \tag{4.5}$$

L'altitude interpolée d'un point p peut être estimée par l'équation suivante :

$$z(p) = \frac{z(C_2)d_A(p,C_1) + z(C_1)d_A(p,C_2)}{d_A(p,C_1) + d_A(p,C_2)} \tag{4.6}$$

avec $z(C_i)$ l'altitude de la courbe de niveau C_i et $d_A(p,C_i)$ la distance géodésique entre p et C_i (cf. figure 4.3).

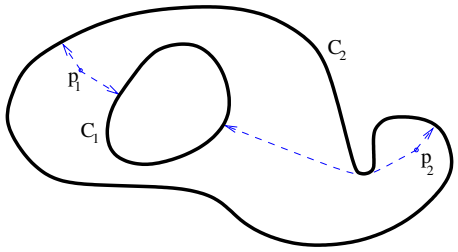


FIG. 4.3: *Distances géodésiques entre les points p_1 et p_2 et les courbes de niveau les plus proches.*

Dans la pratique, cette technique d'interpolation peut s'effectuer à l'aide d'algorithmes de type propagation ou feu de forêt. Par exemple, Soille [Soi91] propose un algorithme rapide

par propagation permettant de calculer une approximation de la transformée en distance géodésique basée sur la métrique Euclidienne sur une grille régulière.

Comme cet algorithme interpole l'altitude et non la pente, le terrain résultant est C^1 partout sauf sur les courbes de niveau et sur les axes médians. Les pentes n'étant pas interpolées, les sommets des montagnes ou les puits sont reconstruits sous la forme de plateaux.

4.2.4 Equations aux Dérivées Partielles

Les EDP sont des équations dont les solutions sont les fonctions inconnues vérifiant certaines conditions concernant leurs dérivées partielles. La résolution de telles équations est généralement complexe, mais peut être approximée à l'aide de méthodes itératives comme la méthode des éléments finis. Dans cette section, nous présentons deux EDP utilisées pour l'interpolation de courbes de niveau [GF98]. La première est une application du modèle Laplacien, la deuxième, un peu plus complexe mais donnant de meilleurs résultats est connue comme le modèle Lame Fine (Thin-plate).

4.2.4.1 Laplacien

Le modèle EDP le plus simple que l'on peut utiliser pour interpoler des altitudes sur une grille est le Laplacien, aussi connu sous le nom de "équation de la chaleur". En 2D, l'équation de cette EDP est :

$$\frac{\partial^2 z}{\partial x^2} + \frac{\partial^2 z}{\partial y^2} = 0 \tag{4.7}$$

Ce modèle permet généralement de modéliser la conduction de la chaleur dans une plaque de métal fine sur laquelle des points sont chauffés à une température fixée. Dans le cas de l'interpolation de courbes de niveau, nous supposons que les points des courbes de niveau sont fixés à une altitude connue et que les autres points doivent converger à une altitude d'équilibre stable.

La résolution numérique de l'équation 4.7 sur une grille régulière peut s'obtenir en utilisant la méthode des éléments finis à l'aide de l'équation suivante :

$$z_{x-1,y} + z_{x+1,y} + z_{x,y-1} + z_{x,y+1} - 4 \cdot z_{x,y} = 0 \tag{4.8}$$

avec $z_{x,y}$ l'altitude du point de coordonnées (x, y). L'évaluation itérative de cette équation pour chaque point de la grille permet de converger vers la solution de l'EDP. Cependant, la convergence vers la solution est très lente et n'est pas assurée. Une technique de relaxation peut alors être utilisée pour accélérer la convergence :

$$\widetilde{z}_{x,y} = \lambda z_{x,y} + (1 - \lambda) {z'}_{x,y} \tag{4.9}$$

avec λ le pas de relaxation choisi entre 1 et 2, $z'_{x,y}$ et $z_{x,y}$ sont respectivement l'altitude calculée à l'itération précédente et l'altitude courante. Différentes techniques peuvent être utilisées pour résoudre le système linéaire surdéterminé impliquant une matrice éparse engendrée par l'équation 4.8. La méthode itérative de Gauss-Seidel est souvent utilisée. Franklin et Childs [Chi03] proposent d'utiliser l'algorithme des moindres carrés (LSQR) de Paige et Saunders's [PS82] avec une solution initiale calculée par quelques milliers d'itérations de la méthode des éléments finis. Une autre solution efficace et rapide donnant de bons résultats est basée

sur une approche "multi-grille", consistant à résoudre le système sur une pyramide de grille, une grille de plus faible résolution étant utilisée comme condition initiale pour la grille de résolution deux fois plus précise, et ainsi de suite.

Comme avec la technique IDWA, le principal artefact généré par le modèle Laplacien est l'effet de terrassement [GF98] survenant entre les courbes de niveau initiales dans les zones où la courbe la plus basse est plus longue que la courbe la plus haute, car la courbe la plus basse contribue plus que la courbe haute (voir figures 4.4). N'interpolant que l'altitude sans prendre en compte sa dérivée (la pente), la surface obtenue avec ce schéma d'interpolation présente une discontinuité au niveau des courbes de niveau et les sommets et puits ne sont pas interpolés et sont modélisés sous la forme de plateaux.

FIG. 4.4: *Effet de terrassement entre deux courbes de niveau.*

4.2.4.2 Spline plaque mince

En 1974, Briggs [Bri74] proposa d'utiliser le modèle d'EDP de plaque mince pour interpoler les courbes de niveau. Ce modèle est une EDP du quatrième ordre venant de la modélisation physique de la forme prise par une fine plaque de métal lorsqu'elle est contrainte à des conditions de bords données. La plaque minimise alors son énergie de courbure :

$$E = \iint \left(\left(\frac{\partial^2 z}{\partial x^2} \right)^2 + 2 \left(\frac{\partial^2 z}{\partial x \partial y} \right)^2 + \left(\frac{\partial^2 z}{\partial y^2} \right)^2 \right) dxdy. \tag{4.10}$$

Sur une grille, l'équation 4.10 peut être approximée, à l'aide de la méthode des éléments finis, par l'équation suivante :

$$\begin{aligned} 20 \cdot z_{x,y} = \quad & 8(z_{x-1,y} + z_{x+1,y} + z_{x,y-1} + z_{x,y+1}) \\ & -z_{x-2,y} + z_{x+2,y} + z_{x,y-2} + z_{x,y+2} \\ & -2(z_{x-1,y-1} + z_{x+1,y-1} + \\ & \quad z_{x-1,y+1} + z_{x+1,y+1}) \end{aligned} \tag{4.11}$$

Les méthodes de résolution de l'équation 4.11 sont les mêmes que pour le modèle Laplacien décrit auparavant. Les résultats obtenus avec ce modèle sont meilleurs qu'avec le Laplacien. En effet, ce modèle interpole également le gradient de la surface, et donc la pente. La surface obtenue est donc C^1 partout et produit donc moins d'effet de terrassement et plus de plateaux.

L'inconvénient est qu'un autre type d'artefact est créé : un effet d'oscillation [GF98], similaire au phénomène de Gibbs dans les séries de Fourier.

Une des meilleures implémentations de cette méthode est disponible dans l'outil TOPOGRID [Hut88] disponible dans le logiciel de SIG, ArcInfo d'ESRI. En plus d'inclure une pénalité de rugosité dans l'équation Thin-plate, la méthode calcule préalablement les lignes de crêtes et les lignes de flux afin d'obtenir un modèle plus précis.

4.2.5 Le krigeage

Le *krigeage* est une méthode d'interpolation stochastique issue de la géostatistique permettant d'interpoler un ensemble de points dans un espace fini. Le nom de krigeage et la notion de géostatistique [JH78] furent introduites au début des années 1970 par Matheron dans ses recherches sur la théorie des variables régionalisées [Mat71]. Matheron formalisa les travaux de Danie G. Krige, un ingénieur de mines Sud-Africain qui avait proposé dans les années 1950 [Kri51] des concepts innovant d'estimation dans le cadre d'explorations minières. Le krigeage est basé sur la variable généralisée relative à un phénomène s'étalant dans l'espace ou le temps. On distingue 3 types de krigeage dépendant de la statistique de la variable à interpoler :

1. le krigeage simple qui nécessite une variable stationnaire de moyenne connue,
2. le krigeage ordinaire, le plus largement utilisé, car il ne nécessite pas de connaissance a priori de la moyenne,
3. le krigeage universel qui ne nécessite pas la stationnarité.

Dans le cadre de l'interpolation de valeurs d'altitude, nous utilisons le krigeage ordinaire.

D'une certaine manière, le krigeage ordinaire est assez similaire au modèle IDWA. En effet, l'estimation d'une valeur se fait en utilisant une combinaison linéaire des valeurs des points connus. En un point p, l'équation de krigeage s'écrit de la façon suivante :

$$z(p) = \sum_{i=1}^{n} \lambda_i z_{p_i} \tag{4.12}$$

avec n le nombre d'échantillons connus et λ_i les poids associés avec valeur tels que $\sum_{i=1}^{n} \lambda_i = 1$. Ces poids sont calculés à partir du semi-variogramme (la moitié du variogramme). Dans notre étude, il peut être modélisé en utilisant le modèle exponentiel suivant :

$$\gamma(d_{p,q}) = C * \left(1 - exp\left(-3\frac{d_{p,q}}{a}\right)\right), \tag{4.13}$$

avec d_{p_1,p_2} la distance entre les points p et q, C la valeur maximum du variogramme et a le rang du variogramme.

Le système du krigeage ordinaire est obtenu en minimisant la variance d'estimation avec la contrainte sur les poids. Dans [Aub96], Aubry montre qu'il est possible d'écrire le krigeage sous sa forme duale de la manière suivante :

$$\begin{cases} z(p) = \sigma + \sum_{i=1}^{n} \mu_i \gamma(d_{p,p_i}) \\ \\ \sum_{i=1}^{n} \mu_i = 0 \end{cases} \tag{4.14}$$

L'équation 4.14 détermine un système linéaire de type $Ax = b$ avec :

$$\begin{aligned} A &= \begin{pmatrix} \gamma(d_{p_1,p_1}) & \dots & \gamma(d_{p_1,p_n}) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ \gamma(d_{p_n,p_1}) & \dots & \gamma(d_{p_n,p_n}) & 1 \\ 1 & \dots & 1 & 0 \end{pmatrix} \\ x &= (\mu_1, \dots, \mu_n, \sigma)^T \\ b &= (z_{p_1}, \dots, z_{p_n}, 0)^T \end{aligned} \tag{4.15}$$

La matrice A est symétrique, mais aussi pleine. Aucune optimisation n'est donc véritablement exploitable pour résoudre ce système. Une solution directe s'effectue avec des complexités de $O(n^3)$ en temps et $O(n^2)$ en mémoire. L'évaluation de l'équation 4.12 pour m points se fait en temps $O(mn)$.

Si toutes les données sont utilisées pour l'estimation d'un point, seul le vecteur b varie en fonction du point testé, alors que la matrice A reste inchangée. Cependant, il n'est généralement pas nécessaire d'utiliser l'ensemble des valeurs initiales pour estimer un point. En effet, seules les valeurs des points proches sont réellement influentes, tandis que les points lointains auront un poids proche de zéro, rendant donc négligeable leur contribution dans la valeur estimée.

Le MNA reconstruit avec la méthode du krigeage est lisse mais souffre d'oscillations autour des échantillons.

4.2.6 Triangulation

La triangulation d'un MNT consiste à générer un maillage de triangle dont les vertex font partis des échantillons disponibles : soit les points cotés épars, soit des points régulièrement échantillonnés sur les courbes de niveau cotées. Les techniques de triangulation ne font pas nécessairement usage de l'interpolation proprement dite.

La technique classique (et disponible dans de nombreux modeleurs 3D comme Autodesk *3ds Max*) consiste à générer une triangulation de Delaunay contrainte sur l'ensemble des échantillons disponibles et en préservant des arêtes entre les échantillons voisins d'une même courbe. Cette technique est extrêmement rapide et simple à mettre en œuvre (en particulier avec la bibliothèque `Triangle` de Shewchuk [She96]) mais souffre de plusieurs défauts. Le principal problème est que des triangles "plats" (ie. dont les vertex ont tous les 3 la même altitude, voir figure 4.5) peuvent apparaître si les 3 points utilisés sont issus de la même courbe de niveau (toujours le cas lors de la triangulation des courbes de niveau les plus élevées). Le second problème est que si les échantillons sont trop espacés, la triangulation obtenue peut être trop grossière pour être exploitée aussi bien en terme de visualisation qu'en terme d'analyse topographique.

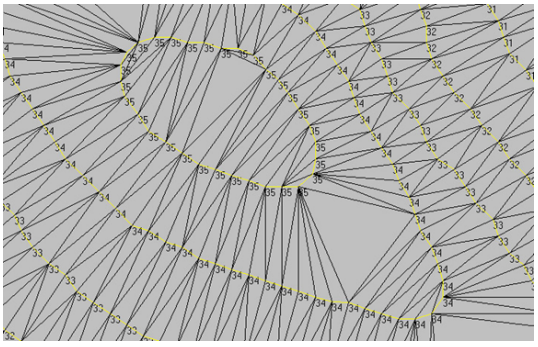


FIG. 4.5: *Triangulation RTI à l'aide de points échantillonnés sur les courbes de niveau.*

Pour régler le problème des triangles "plats", Thibault et Gold [TG00] proposent d'insérer dans la triangulation, les points du squelette extrait avec la méthode d'Amenta [ABE98] et correspondant aux points du diagramme de Voronoï des échantillons disponibles. Les altitudes des points du squelette simplifié après ébarbulage sont estimées en fonction de leur distance

aux 2 courbes voisines. La figure 4.6 montre un exemple de la triangulation obtenue.

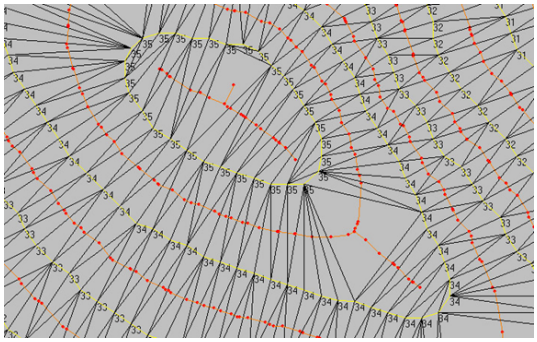


FIG. 4.6: *Triangulation RTI selon [TG00] à l'aide de points échantillonnés sur les courbes de niveau et enrichi avec des points du squelette.*

Dans [HSS03], Hormann *et al.* proposent une méthode pour générer un maillage dont la surface est C^1-continue sauf sur le squelette des courbes utilisées en entrée décrivant les crêtes et vallées du relief. En plus d'insérer dans la triangulation des points du squelette comme dans la méthode de Thibault et Gold, les auteurs proposent, à l'aide de `Triangle`, d'enrichir le maillage avec des points supplémentaires, dits points de Steiner, pour restreindre la taille des triangles obtenus (voir figure 4.7). Les altitudes des points du squelette sont évaluées par interpolation géodésique tandis que les altitudes des points de Steiner sont calculées par interpolation de Hermite en prenant en compte non seulement l'altitude des courbes cotées, mais également la pente en celles-ci. En reproduisant les lignes caractéristiques de crêtes et de vallées, et en prenant également en compte les sommets et les puits, le modèle reconstruit reconstitue un modèle reproduisant relativement bien des zones montagneuses.

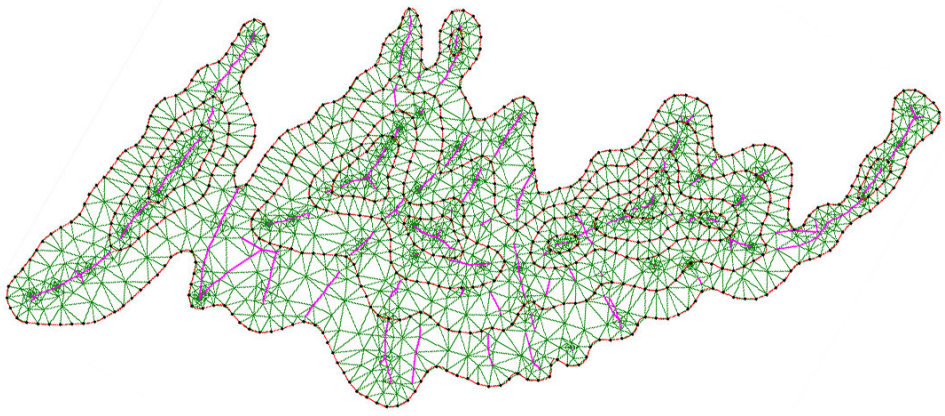


FIG. 4.7: *Triangulation selon [HSS03] enrichie des points du squelette (en rose) et des points de Steiner.*

4.3 Interpolation lisse par Fonctions à Base Radiale

Nous présentons maintenant un algorithme efficace d'interpolation de MNA basé sur les fonctions à base radiale (FBR, en anglais *radial basis functions* ou RBF) associées à une tech-

nique connue sous le nom de partition de l'unité (PdU) permettant de composer des solutions locales pour obtenir une solution globale. Nous proposons une amélioration à la PdU, qualifiée de hiérarchique, permettant d'optimiser la reconstruction et son temps d'évaluation. L'algorithme présenté dans cette section est une adaptation au MNA de la technique développée par Irek Tobor *et al.* [TRS04] et a fait l'objet d'une publication dans la conférence internationale ACM GIS 2004 [PTGG04].

Le schéma de reconstruction global se divise en 3 étapes :

1. division hiérarchique du domaine
2. reconstruction des fonctions d'interpolation locales sur chaque sous-domaine
3. évaluation de la fonction d'interpolation globale

La reconstruction par PdU étant quasi indépendante du partitionnement géométrique, nous proposons un nouveau schéma de partitionnement qui minimise le temps de chacune des deux étapes. Premièrement, il semble clair que le temps minimal de partitionnement est atteint si tous les sous-domaines ont un nombre de points égal. Deuxièmement, le temps d'évaluation de $f(\mathbf{p})$ est plus court si le schéma de partitionnement permet une recherche rapide des domaines contenant le point $\mathbf{p}$. Notre schéma de reconstruction et d'évaluation optimise chacun de ces deux temps de traitement. Ce schéma permet aussi plus de contrôle sur la robustesse et le stabilité de la solution numérique.

Dans la suite de cette section, nous présentons tout d'abord le cadre théorique de l'interpolation par FBR et de la PdU. Ensuite, nous décrivons l'approche hiérarchique ainsi que le schéma d'évaluation que nous proposons. Enfin, après une courte étude de complexité, nous présentons les résultats obtenus avec notre méthode.

4.3.1 Présentation théorique

4.3.1.1 Reconstruction par FBR

Soit un ensemble de n points distincts $\mathcal{P} = \{\mathbf{p}_1, ..., \mathbf{p}_n\}$ de dimension 2 tels que $\mathbf{p}_k = \{\mathbf{p}_k^x, \mathbf{p}_k^y\} \in \mathbb{R}^2$, et l'ensemble de valeurs $\{h_1, \dots h_n\}$. On souhaite trouver une fonction $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ satisfaisant les contraintes suivantes :

$$f(\mathbf{p}_i) = h_i \quad i = 1 \dots n. \tag{4.16}$$

Un tel problème ayant une infinité de solutions, les techniques variationnelles par FBR permettent d'obtenir une solution minimisant une "énergie" ou la "smoothness" de la fonction reconstruite. Duchon [Duc77] a montré que la fonction de lissage la plus simple est une combinaison linéaire de fonctions à base invariantes à la rotation :

$$f(\mathbf{p}) = \sum_{i=1}^{n} \omega_i \phi(||\mathbf{p}, \mathbf{p}_i||) + \pi(\mathbf{p}), \tag{4.17}$$

avec $||\mathbf{p}_i, \mathbf{p}_j||$ la distance Euclidienne entre les points p_i et p_j, ω_i le poids de la combinaison linéaire (à calculer), $\phi : \mathbb{R} \rightarrow \mathbb{R}$ une fonction positive de base définie et π un polynôme de degré m dépendant du choix de ϕ.

Pour la reconstruction de données altimétriques, le meilleur choix proposé par Hardy [Har71] est la fonction multi-quadrique suivante :

$$\phi(r) = \sqrt{r^2 + \alpha^2}, \tag{4.18}$$

où α est un paramètre permettant de favoriser le degré de lissage au lieu de la fidélité aux données. Le polynôme associé π de degré 1 est lui défini ainsi :

$$\pi(\mathbf{p}) = c_0 + c_1\mathbf{p}^x + c_2\mathbf{p}^y. \tag{4.19}$$

L'équation 4.17 permet de définir un système avec $n+3$ inconnues (ω_i et c_0, c_1, c_2) et seulement n équations. On ajoute alors des contraintes naturelles additionnelles pour les coefficients ω_i sont ajoutées afin d'assurer l'orthogonalité, avec :

$$\sum \omega_i = \sum \omega_i \mathbf{p}_i^x = \sum \omega_i \mathbf{p}_i^y = 0. \tag{4.20}$$

Les équations 4.16, 4.17, et 4.20 déterminent le système linéaire suivant :

$$\boldsymbol{A}\mathbf{x} = \mathbf{b} \tag{4.21}$$

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{\Phi} & \boldsymbol{P} \\ \boldsymbol{P^T} & 0 \end{bmatrix} \tag{4.22}$$

$$\boldsymbol{\Phi} = [\phi(||\mathbf{p_i}, \mathbf{p_j}||)]_{\substack{i=1\ldots n \\ j=1\ldots n}}$$

$$\boldsymbol{P} = [1\ \mathbf{p}_i^x\ \mathbf{p}_i^y]_{i=1\ldots n}$$

$$\mathbf{x} = [\omega_1, \omega_2, \ldots, \omega_n, c_0, c_1, c_2]^T \tag{4.23}$$

$$\mathbf{b} = [h_1, h_2, \ldots, h_n, 0, 0, 0]^T \tag{4.24}$$

Le vecteur solution $\mathbf{x}$ est composé des poids ω_i et des coefficients c_i de l'équation 4.17, et représente une solution au problème d'interpolation de l'équation 4.16.

Comme la matrice A est symétrique mais pas compacte, la résolution numérique du système 4.21 se fait en $O(n^3)$ en temps par décomposition LU et en $O(n^2)$ en mémoire. De plus, une évaluation directe de la fonction f sur une grille de m nœuds nécessite $O(nm)$ opérations. Cette complexité n'est pas acceptable quand n dépasse quelques milliers de points, ce qui est généralement le cas dans le problème de la reconstruction de terrains.

4.3.1.2 Partition de l'unité

La PdU repose sur le principe du *diviser pour régner*. Il s'agit donc de diviser le domaine d'intérêt global en sous-domaines plus petits où le problème peut se résoudre efficacement. Plus précisément, le problème global est décomposé en plusieurs locaux plus petits. Les solutions sont ensuite combinées en utilisant des fonctions de pondération qui agissent comme des fonctions de mélange lisses afin d'obtenir la solution globale.

Nous introduisons ici le cas le plus simple de division du problème en 2 sous-domaines, notre approche hiérarchique étant décrite plus loin (section 4.3.2).

Considérons un domaine Ω et divisons-le en deux sous-domaines se chevauchant Ω_1 et Ω_2, avec $\Omega = \Omega_1 \cup \Omega_2$ et $\Omega_1 \cap \Omega_2 \neq \emptyset$. Sur l'ensemble de sous-domaines $\{\Omega_1, \Omega_2\}$, nous construisons une PdU, c'est-à-dire une collection de fonctions non négatives $\{\lambda_1, \lambda_2\}$ de support limité $\text{supp}(\lambda_i) \subseteq \Omega_i$ et avec $\lambda_1 + \lambda_2 = 1$ sur tout le domaine Ω.

Pour chaque Ω_i, un ensemble $\mathcal{P}_i = \{\mathbf{p} \in \mathcal{P} | \mathbf{p} \in \Omega_i\}$ est défini, puis une fonction de reconstruction locale f_i est calculée. La fonction de reconstruction globale f_{pdu} est ensuite définie comme une combinaison des fonctions locales :

$$f_{pdu}(\mathbf{p}) = f_1(\mathbf{p})\lambda_1(\mathbf{p}) + f_2(\mathbf{p})\lambda_2(\mathbf{p}) \tag{4.25}$$

La condition $\lambda_1 + \lambda_2 = 1$ est obtenue pour tout autre ensemble de fonctions lisses Λ_1, Λ_2 à l'aide d'une procédure de normalisation :

$$\lambda_i(\mathbf{p}) = \frac{\Lambda_i(\mathbf{p})}{\Lambda_1(\mathbf{p}) + \Lambda_2(\mathbf{p})} \qquad \text{pour } i = 1, 2 \tag{4.26}$$

Les fonctions de pondération Λ_i doivent être continues à la frontière des sous-domaines Ω_i. Nous définissons les fonctions de pondération Λ_i comme la composition d'une fonction de distance $D_i : \mathbb{R}^n \to [0, 1]$, où $D_i(\mathbf{p}) = 1$ à la frontière de Ω_i et une fonction d'affaiblissement $V : [0, 1] \to [0, 1]$, telle que $\Lambda_i(\mathbf{p}) = V \circ D_i(\mathbf{p})$.

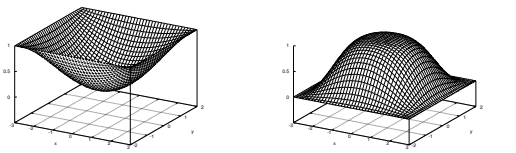
Pour une boîte 2D alignée sur les axes et définie par deux coins opposés S et T nous utilisons la fonction de distance D_i suivante :

$$D_i(\mathbf{p}) = 1 - \prod_{r \in x,y} \frac{4(\mathbf{p}_r - S_r)(T_r - \mathbf{p}_r)}{(T_r - S_r)^2}. \tag{4.27}$$

Le choix de la fonction V détermine la continuité entre les solutions locales f_i dans la fonction de reconstruction globale f_{pdu}. Nous proposons d'utiliser une des fonctions d'affaiblissement suivantes qui ont été choisies en incluant des contraintes similaires à la construction des fonctions de bases Spline ($V(0) = 1$, $V(1) = 0$, $V'(0) = V'(1) = 0$, etc.) :

$$\begin{aligned} &\text{continuité } \mathbb{C}^0 : && V^0(d) = 1 - d \\ &\text{continuité } \mathbb{C}^1 : && V^1(d) = 2d^3 - 3d^2 + 1 \end{aligned}$$

Les tracés de la fonction de distance D et des fonctions de pondération sont illustrés dans la figure 4.8.



(a) $D(\mathbf{p})$ (b) $\Lambda(\mathbf{p}) = (V^1 \circ D)(\mathbf{p})$

FIG. 4.8: *Interprétation de la fonction de distance D et des fonctions de pondération Λ.*

4.3.2 Approche hiérarchique

Notre algorithme de reconstruction peut se décomposer en deux étapes :

1. Le domaine global d'intérêt, c'est-à-dire la grille du MNA est subdivisée récursivement en sous-domaines se chevauchant, chacun contenant un nombre de points connus, quasiment égal. Cette subdivision se fait à l'aide d'une structure appelée k-d arbre (voir section 4.3.2.1). Les régions de chevauchement sont utilisées dans le processus de PdU afin de mélanger les solutions locales.
2. La fonction de reconstruction locale dans chaque feuille est calculée en utilisant la technique variationnelle avec FBR.

L'évaluation de la fonction de reconstruction globale est faite récursivement en propageant et en mélangeant les solutions locales des feuilles jusqu'à la racine (voir section 4.3.2.2). La dernière opération de mélange obtenue à la racine décrit le résultat de la fonction de reconstruction globale.

4.3.2.1 Décomposition binaire du domaine

A partir d'un ensemble de points épars $\mathcal{P}$ et du domaine d'intérêt global Ω^0 (généralement défini par le rectangle englobant de $\mathcal{P}$), la méthode de décomposition en kd-arbre de sous-domaines réalise une subdivision adaptative de Ω^0. L'arbre est construit récursivement du haut vers le bas à partir du nœud racine. Le domaine Ω^0 est ensuite subdivisé en deux sous-domaines Ω^1_1 et Ω^1_2 contenant les ensembles de points $\mathcal{P}^1_1$ et $\mathcal{P}^1_2$ de cardinal respectifs quasi identiques n^1_1 et n^1_2. De façon récursive tous les sous-domaines Ω^l_k au niveau l sont eux-mêmes subdivisés en deux sous-domaines $\Omega^{l+1}_{k,1}$ et $\Omega^{l+1}_{k,2}$ contenant les ensembles de points $\mathcal{P}^{l+1}_{k,1}$ et $\mathcal{P}^{l+1}_{k,2}$ et cardinal respectifs $n^{l+1}_{k,1}$ et $n^{l+1}_{k,2}$. La récursion se termine quand le nombre de points dans un sous-ensemble atteint une limite inférieure $T_{feuille}$.

Nous donnons maintenant quelques détails sur la subdivision du domaine Ω^l_k.

Tout d'abord, le nombre minimum de points $n^l_o = \text{Card}(\mathcal{P}^{l+1}_{k,1} \cap \mathcal{P}^{l+1}_{k,2})$ dans la zone de recouvrement $\Omega^{l+1}_{k,1} \cap \Omega^{l+1}_{k,2}$ doit être spécifiée explicitement comme un *quota de recouvrement* $q \in]0,1[$ de nombre de points n^l_k :

$$n^l_{o,k} = q n^l_k \tag{4.28}$$

Ensuite, les nombres minimum de points n^{l+1}_k dans les sous-domaines peuvent être calculés comme suit :

$$n^{l+1}_k = \left\lceil \frac{n^l_{o,k} + n^l_k}{2} \right\rceil \tag{4.29}$$

L'étendue des deux sous-domaines $\Omega^{l+1}_{k,1}$ et $\Omega^{l+1}_{k,2}$ est calculée en utilisant le principe du k-d arbre. Tout d'abord, l'axe le plus long de Ω^l_k est déterminé. Puis, nous collectons les ensembles de points $\mathcal{Q}^{l+1}_{k,1}$ (respectivement $\mathcal{Q}^{l+1}_{k,2}$) contenant les n^{l+1}_k points avec la plus petite (respectivement grande) valeur en fonction de l'axe le plus long. En pratique, nous réordonnons les points $\mathcal{P}^l_k$ en fonction de leur valeur le long de l'axe le plus long, et nous prenons les premiers (respectivement derniers) n^{l+1}_k points : en affectant $i_1 = n^{l+1}_k$ et $i_2 = n^l_k - n^{l+1}_k + 1$, nous réordonnons les points de façon à ce que$\mathbf{p}_i < \mathbf{p}_{i_1}$ pour $1 \leq \mathbf{p}_i \leq \mathbf{p}_{i_1}$ et $\mathbf{p}_i > \mathbf{p}_{i_2}$ pour $\mathbf{p}_{i_2} \leq \mathbf{p}_i \leq \mathbf{p}_{n^l}$:

$$\mathcal{P}^l_k = \overbrace{\mathbf{p}_1, \ldots, \mathbf{p}_{i_2-1}, \underbrace{\mathbf{p}_{i_2}, \ldots, \mathbf{p}_{i_1}}, \mathbf{p}_{i_1+1}, \ldots, \mathbf{p}_{n^l}}^{\mathcal{Q}^{l+1}_{k,1}} \underbrace{\qquad\qquad}_{\mathcal{Q}^{l+1}_{k,2}} \tag{4.30}$$

Finalement, $\Omega^{l+1}_{k,1}$ et $\Omega^{l+1}_{k,2}$ sont définis par les rectangles englobants de $\mathcal{Q}^{l+1}_{k,1}$ et $\mathcal{Q}^{l+1}_{k,2}$ et les points $\mathcal{P}^{l+1}_{k,1}$, $\mathcal{P}^{l+1}_{k,2}$ comme les sous-ensembles de $\mathcal{P}^{l+1}_k$ qui sont contenus dans $\Omega^{l+1}_{k,1}$, $\Omega^{l+1}_{k,2}$

Il est à noter que $\mathcal{P}^l \rightarrow \mathcal{Q}^{l+1} \rightarrow \Omega^{l+1} \rightarrow \mathcal{P}^{l+1}$ pas sont nécessaires si l'ensemble de points initial $\mathcal{P}$ est un sous-ensemble d'une grille régulière. De plus $\mathbf{p}_{i_2-1}$ et $\mathbf{p}_{i_2}$ de l'équation (4.30) peuvent avoir la même coordonnée sur l'axe de partitionnement.

L'algorithme récursif de décomposition du domaine est donné par l'algorithme 2. Il doit être appelé à partir de ses paramètres initiaux `Decomposer(`$\mathcal{P}$`, `Ω`)`. Finalement, la figure 4.9

illustre les premiers niveaux de l'arbre binaire résultant avec les sous-domaines correspondants.

Require: points $\mathcal{P}$, domaine Ω
Ensure: arbre binaire
  $n = \text{Card}(\mathcal{P})$
  if $n > T_{feuille}$ **then**
    $n_o = qn$
    réarranger les points dans $\mathcal{P}$ selon le plus grand axe de Ω
    déterminer les points $\mathcal{Q}_1$, $\mathcal{Q}_2$
    déterminer les sous-domaines Ω_1, Ω_2
    déterminer les points $\mathcal{P}_1$, $\mathcal{P}_2$
    Decomposer($\mathcal{P}_1$, Ω_1)
    Decomposer($\mathcal{P}_2$, Ω_2)
  else
    calculer la fonction de reconstruction FBR locale pour $\mathcal{P}$
  end if

Algorithme 2 : Decomposer($\mathcal{P}$, Ω)

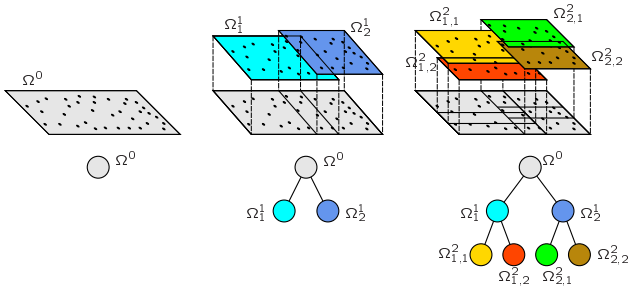


FIG. 4.9: *Premiers niveaux de l'arbre binaire lors de la décomposition en sous-domaines (avec une zone de recouvrement).*

4.3.2.2 Evaluation

Lors de l'évaluation de la fonction de reconstruction globale f pour un point $\mathbf{p}$, les reconstructions locales sont mélangées en appliquant récursivement la méthode de PdU de bas en haut dans l'arbre binaire calculé dans l'étape précédente de décomposition. A partir du nœud racine, à chaque nœud interne de niveau l de l'arbre binaire, la fonction de reconstruction f^l est donnée par le mélange par PdU des fonctions de reconstruction locales des deux nœuds fils f_1^{l+1} et f_2^{l+1} :

$$f^l(\mathbf{p}) = \frac{f_1^{l+1}(\mathbf{p})\Lambda_1^{l+1}(\mathbf{p}) + f_2^{l+1}(\mathbf{p})\Lambda_2^{l+1}(\mathbf{p})}{\Lambda_1^{l+1}(\mathbf{p}) + \Lambda_2^{l+1}(\mathbf{p})} \tag{4.31}$$

L'algorithme récursif 3 illustre l'évaluation de la fonction globale en un point $\mathbf{p}$ et doit être appelée par `Evaluer(p, 0)`.

4.3.3 Analyse de complexité

Comme nous l'avons précédemment indiqué, la solution du système linéaire de l'équation 4.21 de taille N nécessite $O(N^3)$ opérations flottantes et $O(N^2)$ cellules mémoire. Pour une

Require: point $\mathbf{p}$, niveau l
Ensure: $f(\mathbf{p})$
  if $\mathbf{p} \notin \Omega^l$ **then**
    retourner 0
  end if
  if l est un niveau feuille **then**
    retourner $f_{fbr}(\mathbf{p})$
  else
    $f_1 =$ Evaluer($\mathbf{p}$, $l+1$) sur le fils gauche
    $f_2 =$ Evaluer($\mathbf{p}$, $l+1$) sur le fils droit
    calculer Λ_1, Λ_2 pour les fils
    retourner $f = (f_1\Lambda_1 + f_2\Lambda_2)/(\Lambda_1 + \Lambda_2)$
  end if

Algorithme 3 : Evaluer($\mathbf{p}$, l)

évaluation de la fonction FBR la résolution directe nécessite $O(N)$ opérations.

Dans notre schéma de reconstruction, la hauteur de l'arbre binaire est $H = \lceil \log(N/T_{feuille}) \rceil$, le nombre de feuilles est $L = \lceil N/T_{feuille} \rceil$ et chaque feuille contient au plus $T_{feuille}$ contraintes. La reconstruction est composée d'une étape de subdivision nécessitant $O(N \log N)$ opérations en virgule flottante, et d'une étape de reconstruction par FBR en temps $O(LT^3_{feuille}) = O(L) = O(N)$.

L'évaluation de $f(\mathbf{p})$ peut se faire en temps borné du fait du nombre limité de sous-domaines comprenant le point $\mathbf{p}$ et dans lesquels l'évaluation de la fonction FBR locale se fait en temps constant.

4.3.4 Résultats et discussion

Pour illustrer notre méthode, nous présentons deux exemples. Le premier met en jeu un MNA obtenu par télé-acquisition (*remote sensing*) que nous avons sous-échantillonné, le second est un MNA initialisé par un ensemble de courbes de niveau. Dans les tableaux suivants, nous appelons #feuilles le nombre de feuilles de l'arbre, t_{arbre} le temps de création du kd-arbre, t_{rec} le temps de reconstruction local (c.-à-d. reconstruction de la FBR dans chaque feuille), t_{eval} le temps d'évaluation de la fonction d'interpolation globale sur l'ensemble des nœuds de la grille du MNA, et t_{total} le temps d'interpolation global d'interpolation, tous les temps étant exprimés en secondes. Pour tous nos tests, nous avons fixé empiriquement le taux de recouvrement à 20%. Il est clair que si ce taux est trop faible, des artefacts peuvent apparaître sur les frontières des zones ; si le quota est trop grand, le MNA obtenu devient trop lisse.

Tous les résultats présentés ont été obtenus sur une station PC équipée d'un Intel Pentium IV cadencé à 3GHz et doté de 1.5Go de mémoire.

4.3.4.1 Mont Washington

Notre premier test repose sur le MNA du Mont Washington produit par l'USGS. Le MNA consiste en une grille de $932 \times 1384 = 1,289,888$ points, illustré par la figure 4.11(a) et 4.11(b)). Les altitudes de la région varient entre 170 et 1913 mètres.

Le modèle original a été sous-échantillonné de façon aléatoire à $43,419$ points, soit $3,3\%$ du modèle original (voir figure 4.11(c)). Les données restantes sont utilisées pour interpoler le MNA sur la grille initiale à l'aide de notre technique d'interpolation hiérarchique par FBR.

Nous effectuons différentes reconstructions en faisant varier le paramètre $T_{feuille}$ décrivant le nombre maximal de points dans les feuilles de l'arbre. Enfin, les MNA reconstruits sont comparés au modèle original à l'aide de la RMSE (racine carrée de l'erreur quadratique moyenne, en anglais *Root Mean Square Error*). Les résultats numériques obtenus en utilisant un facteur de lissage $\alpha = 2$ (à noter que la valeur de α n'influence en rien le temps de calcul) avec notre implémentation sont décrits dans la table 4.1.

TAB. 4.1: *Résultats numériques pour le MNA du Mont Washington, 43419 points de données (sur les 1289888 de la grille) avec* $\alpha = 2$

$T_{feuille}$	#feuiles	t_{arbre}	t_{rec}	t_{eval}	t_{total}	RMSE
1600	128	22	854	188	1066	5.04
800	256	20	374	157	531	5.05
600	512	16	170	100	287	5.05
200	2048	15	32	59	108	5.06
100	4096	18	16	46	81	5.09

Nous constatons que pour des valeurs très différentes de $T_{feuille}$, les valeurs de RMSE restent assez proche. Ceci est du au fait que les points de données sont dispersés assez régulièrement sur la grille et dans ce cas, le résultat est assez indépendant du choix de la valeur de $T_{feuille}$.

Dans un deuxième temps, le modèle original est sous-échantillonné, toujours de façon aléatoire, en un ensemble de points variant entre 1% à 20% du modèle original. Pour chaque MNA, la fonction de reconstruction est calculée (avec $T_{feuille} = 100$ et $\alpha = 2$), le MNA est ensuite interpolé sur toute la grille et l'erreur RMSE est calculée. La figure 4.10 confirme la complexité théorique en terme de temps de notre algorithme. On note également que la RMSE décroît tandis que le nombre de points de données augmente.

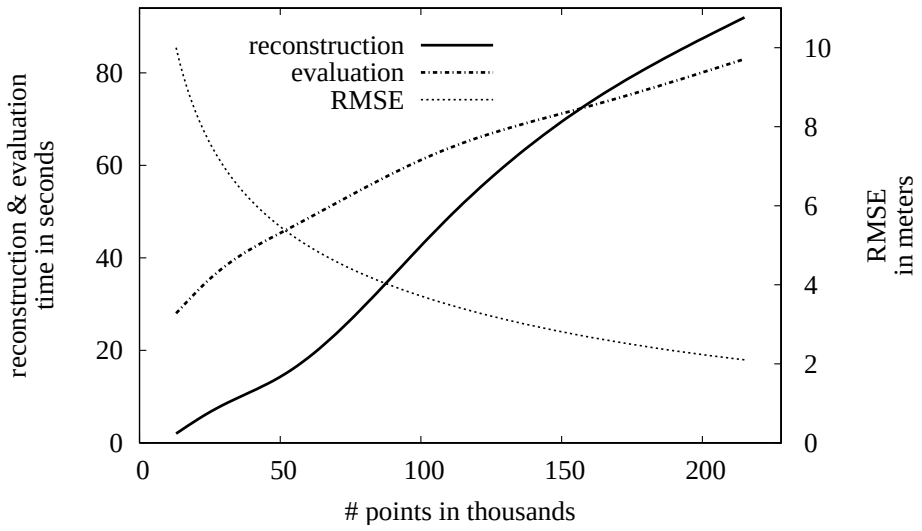


FIG. 4.10: *Temps de calcul et erreur RMSE.*

La figure 4.11(d) montre une vue 3D du modèle de terrain reconstruit avec les paramètres $\alpha = 5$ et $T_{feuille} = 800$. On constate que la plupart des caractéristiques du terrain sont

préservées quand bien même le terrain est lisse.

4.3.4.2 Mont Saint Helens

Dans ce second test, nous utilisons un ensemble de courbes de niveau représentant le volcan du mont Saint Helens, situé dans l'état de Washington, USA, après son éruption le 18 mai 1980.

La carte de contours est constituée de 21 courbes, rastérisées sous la forme de 21442 points de données sur une grille de taille 449×497, soient 223153 points (voir figure 4.14). L'altitude relative des courbes varie entre 20 et 740 mètres. La table 4.2 montre les résultats numériques obtenus pour cette base de tests.

TAB. 4.2: *Résultats numériques pour le MNA comportant les courbes de niveau du Mont St Helens. Le MNA est constitué de 21,442 points de données (sur une grille de* $223,153$ *points)*

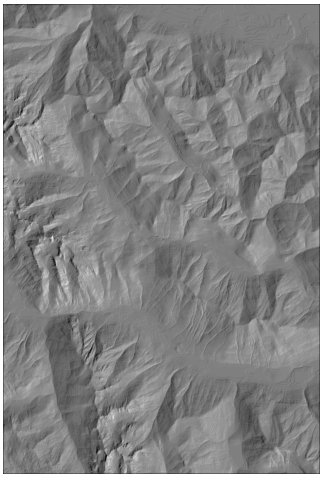
$T_{feuille}$	#feuilles	t_{arbre}	t_{rec}	t_{eval}	t_{total}
1200	32	4	89	13	107
800	64	2	41	10	53
400	128	3	17	7	28

Les figures 4.12(b) et 4.12(c) montrent les modèles interpolés avec coefficients de lissage et $T_{feuille} = 800$. Avec $T_{feuille} = 100$, le modèle interpolé est plutôt acceptable, mais sur la figure 4.12(d), on constate sur la gauche, que certaines zones sont plates et n'ont pas été correctement reconstruites. Ceci est dû au fait que certaines zones des feuilles contiennent plusieurs points avec la même altitude. Il convient donc de choisir un paramètre $T_{feuille}$ suffisamment grand.

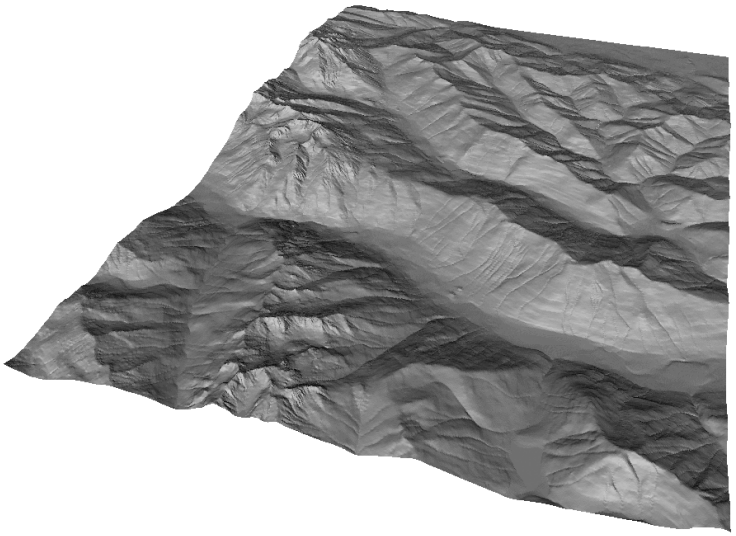
Nous proposons ensuite de sous-échantillonner les données des courbes de niveau. Seuls $2,332$ points de données sur les $21,442$ sont conservés, soit environ 10% (voir figure 4.13(a)). Les temps de calcul obtenus sont résumés dans la table 4.3 et la figure 4.13(b) illustre le modèle de terrain obtenu avec $\alpha = 2$ et $T_{feuille} = 800$. En comparaison avec la figure 4.12(b), le terrain obtenu conserve parfaitement l'apparence globale et la plupart des détails du modèle précédent alors que nous avons utilisé 10 fois moins de données. Aucune oscillation n'est observée entre les échantillons, et c'est encore plus vrai avec des valeurs de α plus grandes. Cela signifie que ce type de sous-échantillonnage n'est pas problématique dans le contexte de la reconstruction d'un MNA à partir de courbes de niveau, et permet d'accélérer grandement le processus.

TAB. 4.3: *Résultats numériques pour le Mont Saint Helens issu de 21 courbes de niveau sous-échantillonnées à* $2,332$ *échantillons (la grille fait* $223,153$ *points au total).*

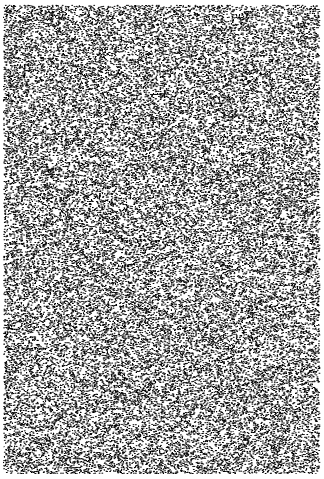
$T_{feuille}$	#feuilles	t_{arbre}	t_{rec}	t_{eval}	t_{total}
1200	4	0	8	7	16
800	8	0	3	5	9
400	16	0	1	3	5



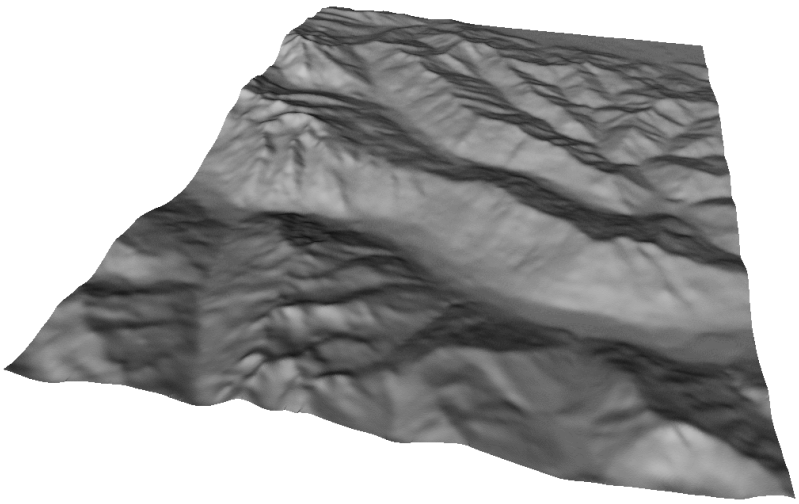
(a) MNA original avec 1314992 points.



(b) Vue 3D du MNA original

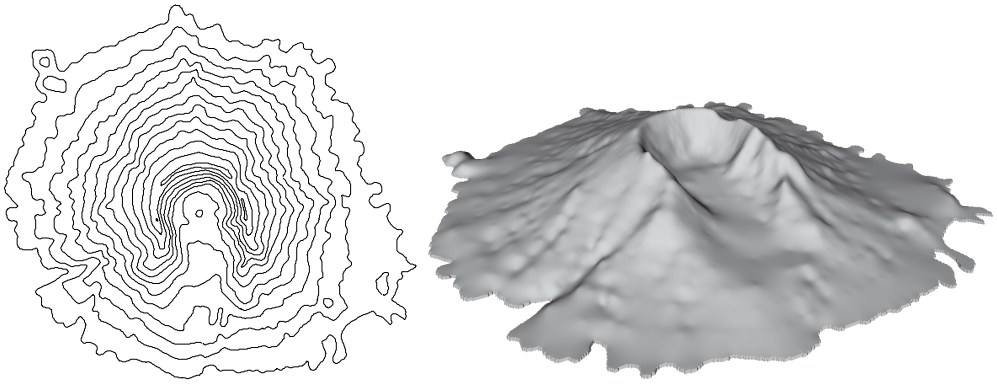


(c) MNA échantillonné avec 43,419 points (3.3% du modèle initial)



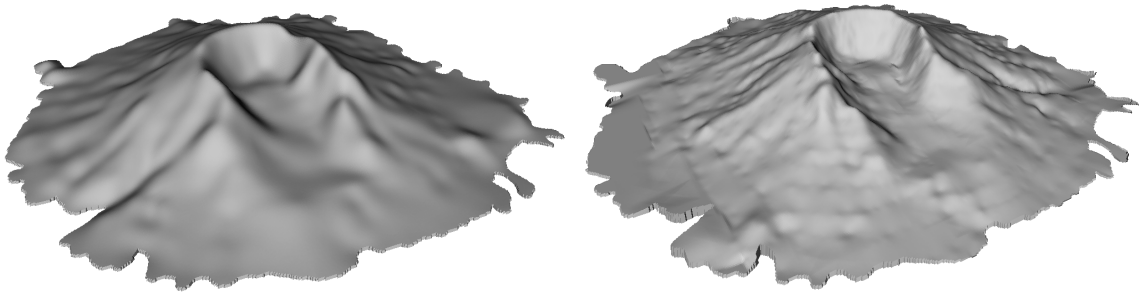
(d) Vue 3D du MNA interpolé avec $T_{feuille} = 800$ et $\alpha = 5$. $t_{total} = 531s$.

FIG. 4.11: *Interpolation du MNA du mont Washington à partir d'échantillons épars.*



(a) Courbes de niveau originales rastérisées sous la forme de 21442 points sur le MNA.

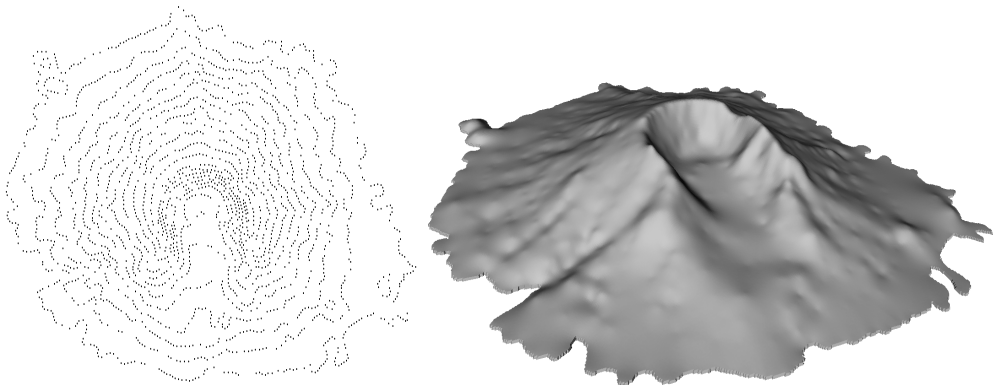
(b) Vue 3D du MNA interpolé avec $\alpha = 2$, $T_{feuille} = 800$ et $t_{total} = 53s$.



(c) $\alpha = 10$ et $T_{feuille} = 800$. $t_{total} = 53s$.

(d) $\alpha = 2$ et $T_{feuille} = 100$. $t_{total} = 10s$.

FIG. 4.12: *Interpolation du MNA du Mont Saint Helens à partir de courbes de niveau.*



(a) Courbes sous-échantillonnées sous la forme de 2,332 points cotés.

(b) $\alpha = 2$ et $T_{feuille} = 800$. $t_{total} = 9s$

FIG. 4.13: *Interpolation du mont Saint Helens à partir des courbes de niveau rastérisées sous-échantillonnées.*

4.4 Expérimentations

Dans cette section, nous décrivons les résultats que nous avons obtenus avec notre implémentation de la plupart des méthodes précédemment décrites. Notre comparaison s'appuie sur des résultats numériques en termes de temps de calcul et d'erreur cumulée par rapport à un MNA obtenu par des techniques de télé-acquisition à partir duquel nous avons extrait les courbes de niveau. Nous montrons ensuite les résultats visuels obtenus à partir d'un cas d'étude.

4.4.1 Résultats numériques

Comme première expérience, nous utilisons le MNA du Mont Saint Helens après son éruption du 18 mai 1980. Il consiste en une grille de taille 951×898 (=853,998 points au total, voir figure 4.14(a)). L'altitude de cette zone varie entre 1300 et 2549 mètres.

Nous avons extrait de ce modèle un ensemble les courbes de niveau avec des équidistances différentes : 25m, 50m, 100m et 200m (voir figure 4.14). Le nombre total de points des courbes de niveau une fois rastérisées sur le MNA sont respectivement 109,677 points, 56,219 points, 33,324 points, et 18,549 points. Pour les techniques d'interpolation par krigeage et FBR seuls 10% de ces points sont conservés.

La figure 4.15 montre les temps de calcul des algorithmes testés obtenus avec un PC Pentium4 cadencé à 3GHz et disposant de 1Go de RAM.

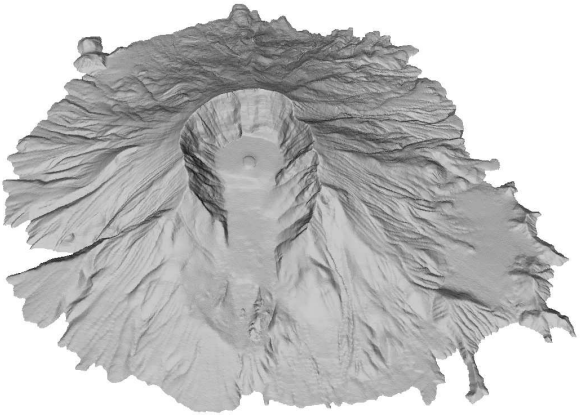
L'erreur *Root Mean Square Error* (RMSE) entre le MNA interpolé et le MNA original est calculée par l'équation suivante :

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(z_i - u_i)^2}, \tag{4.32}$$

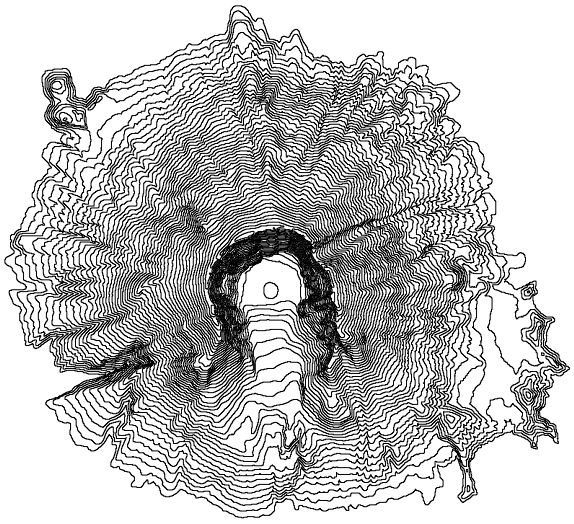
avec u_i l'altitude du MNA original au point i. Les résultats obtenus sont illustrés dans la figure 4.16.

Comme attendu, on constate que plus l'équidistance est grande et que donc moins il y a de contraintes, plus l'erreur RMSE est importante. On constate également que toutes les méthodes implémentées sont quasiment équivalentes en terme de RMSE. Pour le temps de calcul, les interpolations géodésiques, IDWA et EDP sont constantes, car leur complexité dépend principalement de la taille de la grille. Les modèles de krigeage et de FBR sont eux basés sur l'inversion de grandes matrices fonction du nombre de contraintes. Le temps de calcul augmente donc avec la quantité de contraintes initiales.

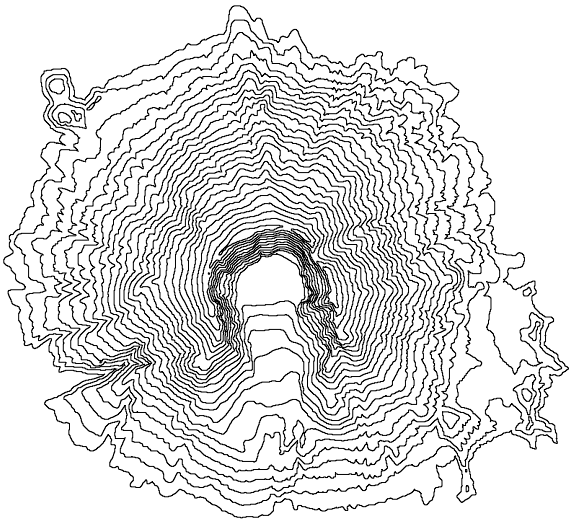
Si l'erreur RMSE ne suffit pas pour différencier la qualité des modèles obtenus, le critère visuel permet quant à lui d'apprécier



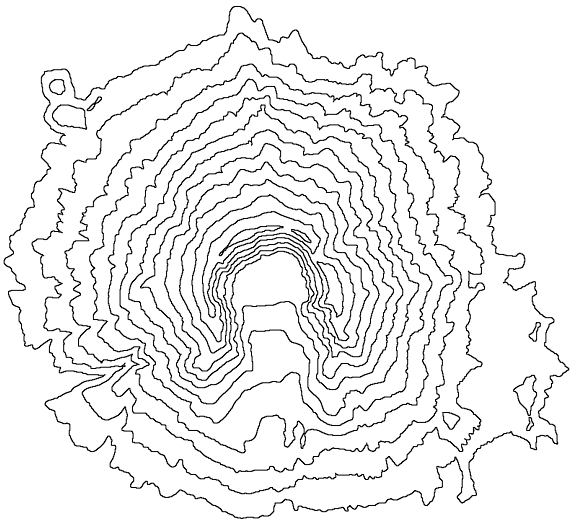
(a) MNA du Mont St. Helens



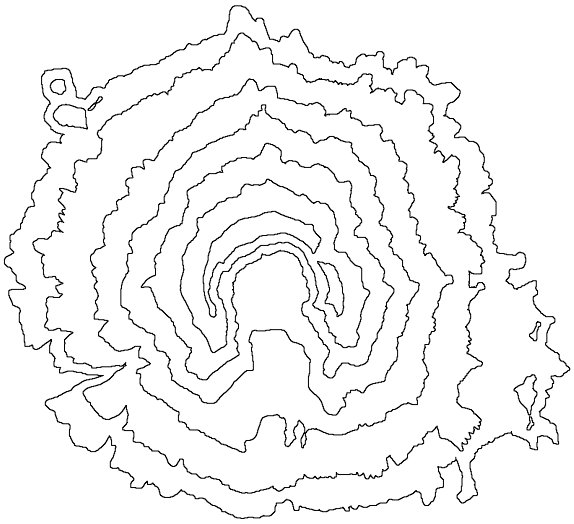
(b) 25m



(c) 50m



(d) 100m



(e) 200m

FIG. 4.14: *Courbes de niveau extraites du MNA de l'USGS autour du Mont Saint Helens.*

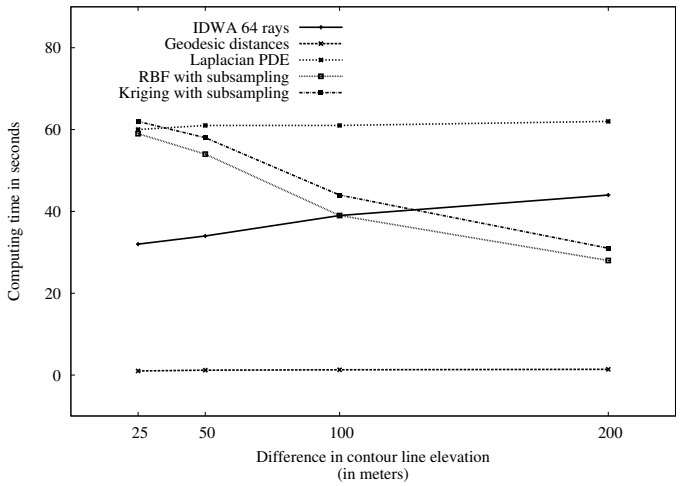


FIG. 4.15: *Temps de calcul pour différents ensembles de courbes de niveau Mont Saint Helens.*

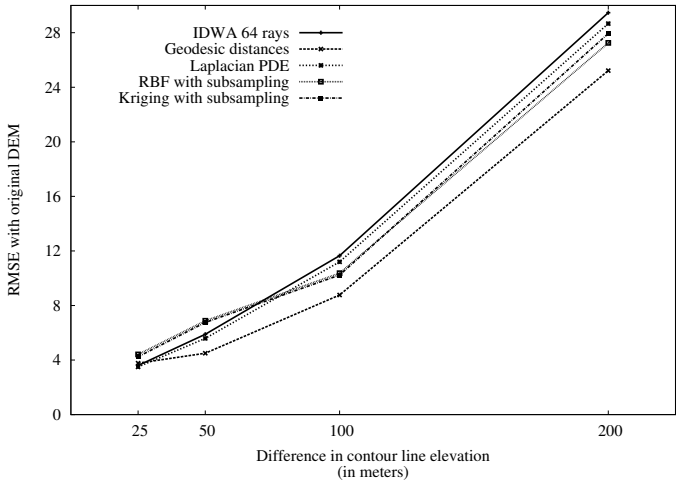


FIG. 4.16: *Erreurs RMSE pour différents ensembles de courbes de niveau Mont Saint Helens.*

4.4.2 Résultats visuels

Dans la mesure où la surface terrestre est tout sauf uniforme en terme de caractéristiques géomorphométriques, il est difficile d'évaluer la crédibilité d'un terrain en termes mathématiques. L'analyse visuelle est certainement le meilleur critère pour juger de la qualité d'un modèle donné. Un test comparatif visuel des différentes techniques présentées est illustré dans la figure 4.17(a). Le modèle initial est constitué de 13 courbes de niveau sur un MNA de taille 737×824. Les courbes de niveau sont rastérisées sous la forme de 7916 points. Pour le krigeage et l'interpolation par FBR, les courbes de niveau ont été sous-échantillonnées sous la forme de 1114 points épars.

La figure 4.17 montre des vues 3D des modèles reconstruits.

On constate que les méthodes IDWA, géodésiques et Laplacien produisent des artefacts à proximité des courbes de niveau à cause d'un manque de continuité de leur fonction d'interpolation. Des plateaux sont générés dans les régions définies par des courbes n'englobant pas d'autres courbes. Pour le krigeage et les FBR, on constate des résultats assez proches et une surface plus lisse du fait de la continuité C^1. La figure 4.18 montre une vue du Mont Saint Helens obtenue avec notre méthode de FBR hiérarchique. Une portion du terrain est agrandie en haut à droite de chaque image et une égalisation de l'histogramme a été appliquée sur chacune de ces vignettes pour mieux mettre les différences en évidence.

Finalement, de notre étude, nous pouvons distinguer deux familles de méthodes pour un usage à fin de visualisation :

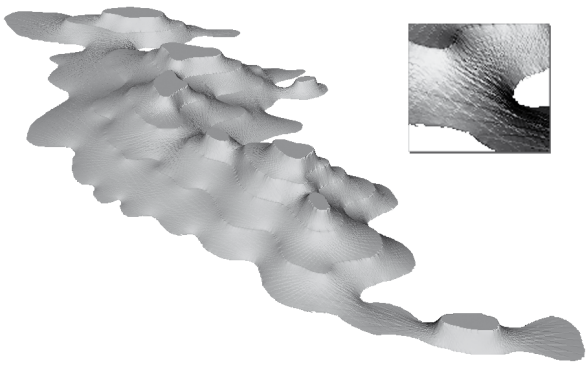
- la première tend à reconstruire un modèle de terrain lisse (c'est le cas des interpolations plaque mince, krigeage et FBR) adapté pour une région peu montagneuse ou constituée de montagnes anciennes érodées ;
- la seconde famille (dont l'interpolation Géodésique ou la triangulation RTI C^1) permettent de reproduire des caractéristiques morphologiques importantes telles que les lignes crêtes en s'appuyant sur l'axe médian. Les modèles ainsi obtenus sont bien adaptés pour la reproduction de zones montagneuses.

4.5 Bilan

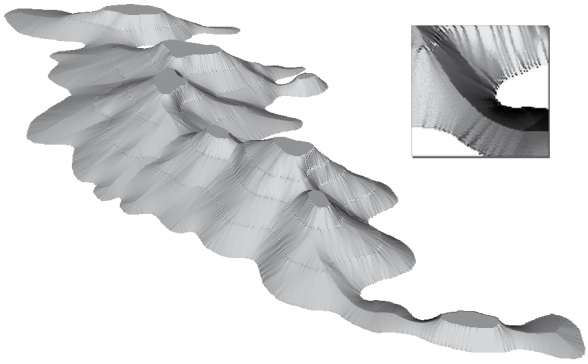
Dans ce chapitre, nous avons présenté les méthodes d'interpolation permettant de générer un MNT à partir de courbes de niveau ou points cotés. La table 4.4 résume les avantages et inconvénients de chaque méthode évaluée.

Nous avons cependant proposé une méthode efficace pour reconstruire un MNA à l'aide des fonctions de base radiale. Les FBR, garanties comme étant l'interpolant le plus lisse tout en atteignant les contraintes initiales, mais impraticables de par leur complexité en temps de calcul, sont utilisées via un schéma de décomposition hiérarchique.

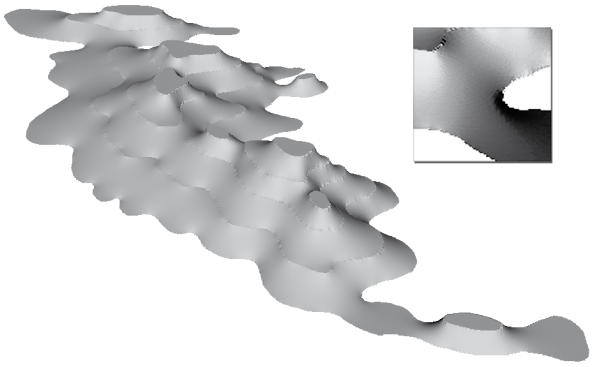
(a) Courbes de niveau initiales.



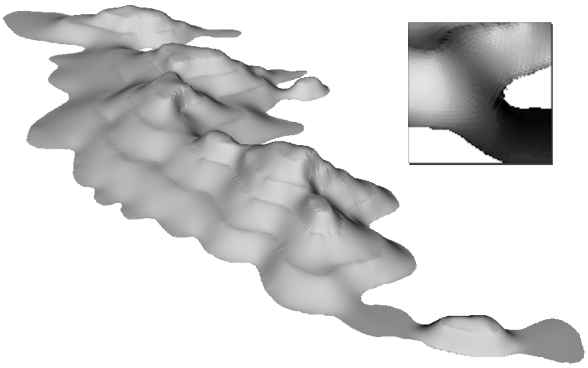
(b) IDWA avec 64 rayons



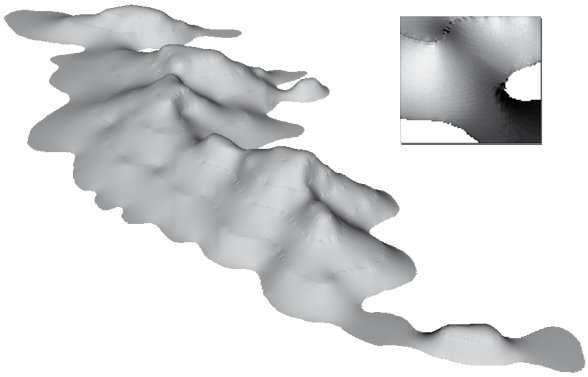
(c) Géodesic



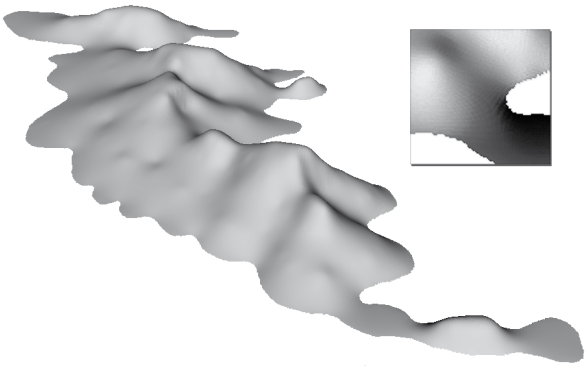
(d) EDP Laplacien avec 50.000 itérations



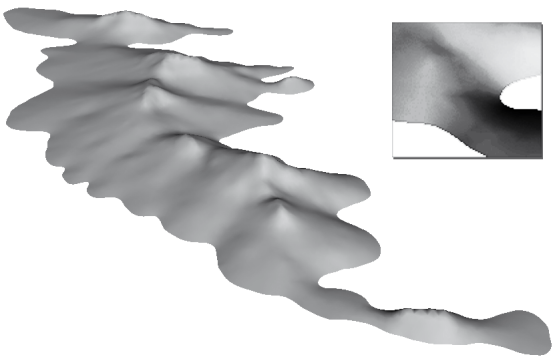
(e) EDP Plaque mince avec 100.000 itérations



(f) Kriging

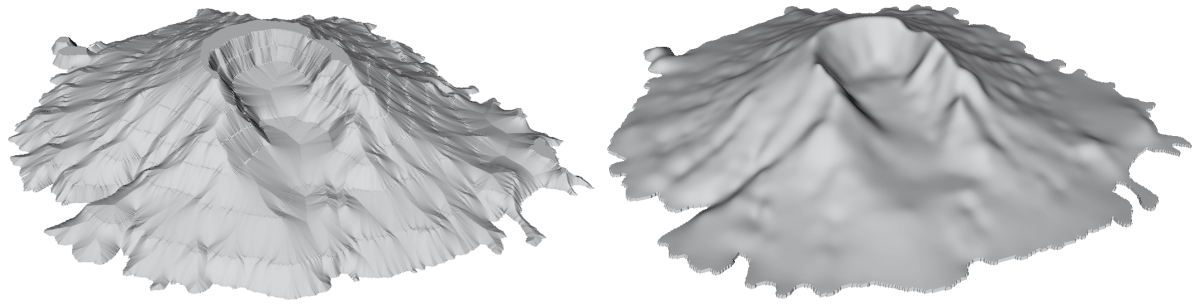


(g) RBF hiérarchiques



(h) C^1 RTI

FIG. 4.17: *Résultats visuels.*



(a) Interpolation géodésique (b) Interpolation hiérarchique par RBF

FIG. 4.18: *Exemple d'interpolation des courbes de niveau du Mont Saint Helens après l'éruption de 1980.*

TAB. 4.4: *Avantages et inconvénients des différentes méthodes.*

Méthode	Données éparses	Implémentation	Lissage	Temps de calcul	Qualité visuelle
IDWA	Y	⋆ ⋆ ⋆ ⋆	Pas lisse	⋆ ⋆	⋆
Géodesique	N	⋆ ⋆ ⋆	Lisse excepté sur les courbes et l'axe médian	⋆ ⋆ ⋆ ⋆	⋆ ⋆ ⋆
Laplacien	Y	⋆ ⋆ ⋆	Lisse excepté sur les courbes	⋆ ⋆	⋆
Plaque-mince	Y	⋆ ⋆ ⋆	Lisse	⋆ ⋆	⋆ ⋆
Krigeage	Y	⋆ ⋆	Lisse	⋆ ⋆	⋆ ⋆ ⋆
FBR hiérarchiques	Y	⋆ ⋆	Lisse	⋆ ⋆ ⋆	⋆ ⋆ ⋆
RTI C^1	N	⋆	Lisse excepté sur l'axe médian	⋆ ⋆ ⋆	⋆ ⋆ ⋆

Chapitre 5

Développement logiciel : la plate-forme AutoDEM

5.1 Introduction

Nous l'avons vu, l'extraction et la numérisation d'information à partir de cartes numérisées est une tâche qui requiert toujours de nombreuses interventions de l'utilisateur. Or s'il existe quelques logiciels commerciaux permettant de réaliser certaines parties des traitements, il n'existe pas de solution complète et extensible spécialisée dans cette tâche. `AutoDEM` est un logiciel de type SIG que j'ai développé pour combler ce manque et nous l'utilisons comme une plate-forme logicielle aux travaux de recherche que nous avons effectué dans ce domaine.

Le logiciel consiste en une interface graphique offrant une palette d'outils et de filtres adaptés à l'extraction manuelle ou semi-automatique et à la conversion des différentes informations extraites. De nombreux formats de fichiers sont supportés, permettant ainsi de traiter les informations extraites dans les logiciels SIG les plus répandus. `AutoDEM` est actuellement distribué gratuitement (hors code source) pour les systèmes Windows.

Le logiciel `AutoDEM` (`AutoMNT` à l'époque) [AutoDEM] a été présenté sous forme de *sketch* à la conférence internationale ACM SIGGRAPH 2004 [PGGG04].

5.2 Architecture

La version actuelle 1.6.2 d'`AutoDEM` est le fruit de nombreuses corrections et réécritures du logiciel `AutoMNT`. Développée en 2003, dans le cadre de mon DEA, la première version était un logiciel assez rigide (bien que des fonctionnalités pouvaient être ajoutées par le biais de plug-ins) et ne permettait que de travailler sur des couches rasters. Quatre ans plus tard, le projet a acquis une certaine maturité et repose désormais sur une architecture beaucoup plus souple et structurée.

5.2.1 Noyau ou SDK

`AutoDEM` représente actuellement plus de 200.000 lignes de code. Il est bâti autour d'une bibliothèque de fonctions et classes écrites en C++ qui en constitue le noyau. Ce noyau étant également le socle sur lequel reposent les fonctionnalités additionnelles, il tient lieu également

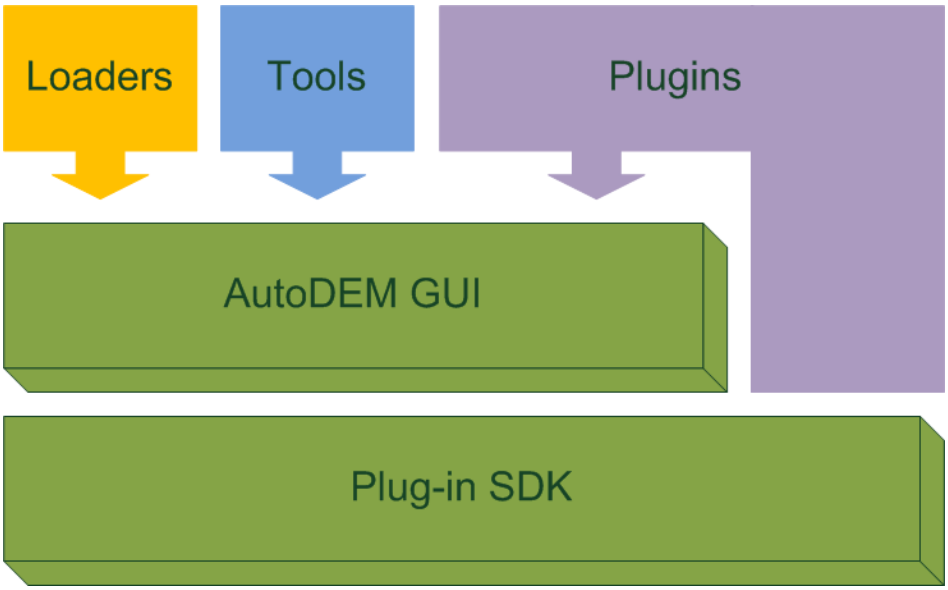


FIG. 5.1: *Architecture globale d'`AutoDEM`.*

de kit de développement (SDK).

Le SDK décrit l'ensemble des structures de données de base (par ex. couches de données, courbes de niveau, toponymes, etc.) indépendantes de l'interface graphique du logiciel. Elle renferme également une grande bibliothèque de fonctions pouvant être utilisées lors du développement de fonctionnalités liées au domaine d'`AutoDEM` (par ex. création d'une couche raster, opérateurs de morpho-mathématiques, conversion de modèles de couleurs, etc.)

5.2.2 Interface utilisateur

L'interface utilisateur du logiciel est basée sur les *Microsoft Foundation Classes* (MFC), bibliothèque de classes C++ dédiées à la mise en œuvre d'applications graphiques sur plate-forme Windows. L'interface du logiciel existe actuellement en français et en anglais.

L'interface utilisateur d'`AutoDEM` (figure 5.2) est assez similaire à tout logiciel de SIG. A chaque projet ouvert (l'interface étant de type multi-documents MDI), est associé un ensemble de couches d'informations. Dans la version 1.6 du logiciel, les couches supportées sont les suivantes :

- couche carte (type raster) ;
- couche de travail (type raster) ;
- couche de courbes de niveau (type vecteurs) ;
- couche de MNA (type raster) ;
- couche de toponymes (type vecteurs) ;
- couche de vecteurs ;
- couche de maillage 3D.

L'utilisateur se voit proposer différents outils pour manipuler chacune de ces couches (les charger, les sauvegarder, en obtenir les propriétés, l'affichage ou non, la transparence, etc.), tandis que des palettes d'outils lui permettent de les modifier ponctuellement (par ex. tracé de lignes, création d'une courbe de niveau, déplacement d'un toponyme, etc.) Les

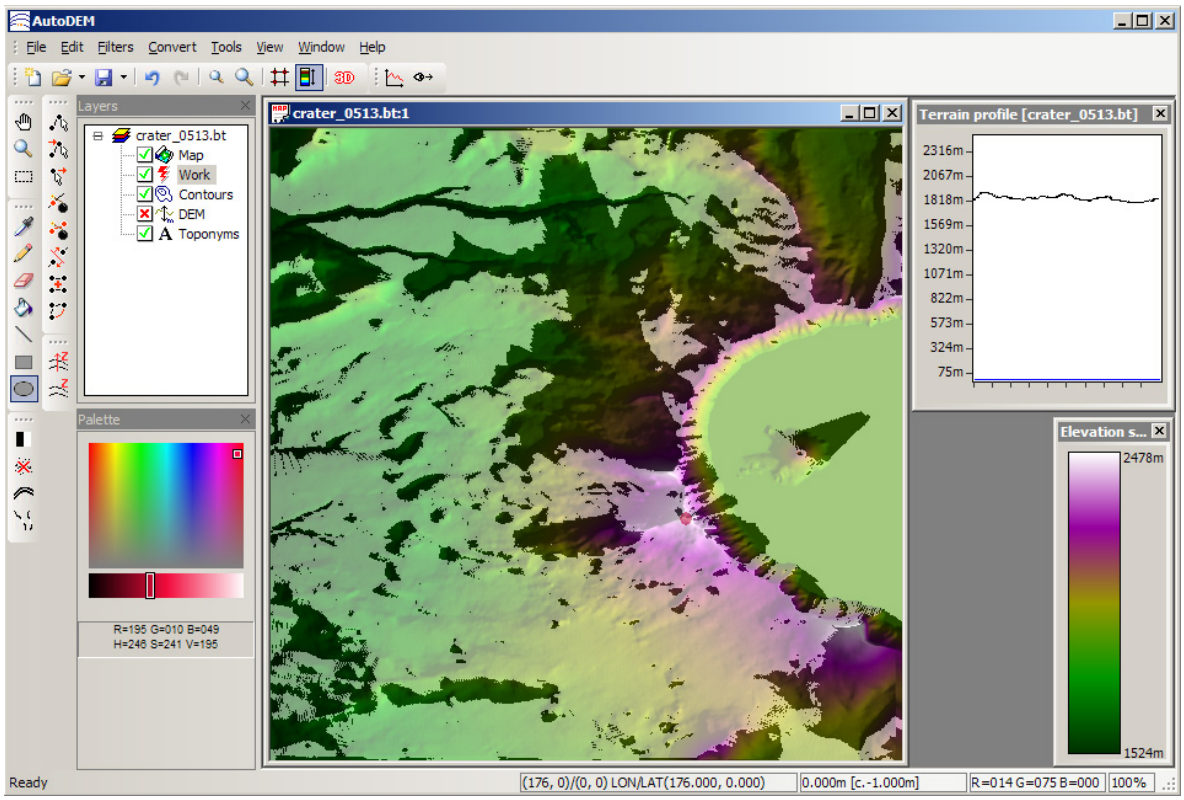


FIG. 5.2: *Interface utilisateur du logiciel* `AutoDEM`.

fonctionnalités les plus intéressantes sont les filtres qui permettent d'appliquer un traitement à une ou plusieurs couches (par ex. application d'une segmentation couleur sur la couche de travail), ou de transformer l'une dans l'autre (par ex. rasterisation de la couche courbes de niveau sur la couche du MNA).

Chaque opération aboutissant à une modification d'une couche est réversible par le biais d'un gestionnaire d'actions annuler / refaire.

5.2.3 Extensibilité par *plug-ins*

`AutoDEM` est un logiciel qui a été conçu comme un outil extensible, que ce soit en terme d'interface ou en terme de fonctionnalités. Cette extensibilité est basée sur le principe de *plug-ins*. Ces *plug-ins* sont des bibliothèques dynamiques (fichier DLL) et sont chargées automatiquement au démarrage du programme. Les *plug-ins* peuvent ajouter 3 types de fonctionnalités au logiciel. Les *chargeurs* permettent de prendre en compte de nouveaux formats de fichiers. Les *outils* permettent d'offrir à l'utilisateur de nouveaux outils (manuels) pour modifier ponctuellement une ou plusieurs couches du projet. Enfin, les *filtres* permettent de définir de nouvelles techniques de traitement des couches.

Chargeurs Dans `AutoDEM`, les chargeurs sont des classes C++ permettant de prendre en compte des formats de fichiers pour une ou plusieurs couches de données que ce soit en lecture ou en écriture. Par défaut, le logiciel fournit de nombreux chargeurs permettant de supporter un grand nombre des formats (offrant ou non la géoréférenciation) classiquement

utilisés dans le domaine des SIG. En particulier, un chargeur permet de prendre en compte les formats supportés par la bibliothèque GDAL (*Geospatial Data Abstraction Library*) [GDAL] développée principalement par Frank Warmerdam. GDAL offre une couche d'abstraction très efficace pour une soixantaine de formats de fichiers couramment utilisés en géomatique (parmi lesquels GeoTIFF, JPEG, PNG, SRTM, USGS DEM, etc.).

`AutoDEM` permet également de décrire des classes d'importation de données. L'importation se distingue ici du chargement dans le sens où elle ne repose pas sur l'ouverture d'un fichier local. Outre la possibilité d'importer une image numérisée directement avec un scanner, nous avons mis au point des classes d'import permettant de supporter des protocoles de type service Internet. L'import le plus intéressant est celui qui permet d'accéder à des bases de données géographiques disponibles au travers du protocole *Web Map Service* (WMS) définit par l'*Open Geospatial Consortium* [OGC].

WMS est un protocole basé sur les protocoles Internet HTTP et URL permettant d'obtenir des informations géoréférencées de type raster sur un serveur géographique. La figure 5.3 montre une capture d'écran de l'interface utilisateur de l'importateur WMS dans `AutoDEM`.

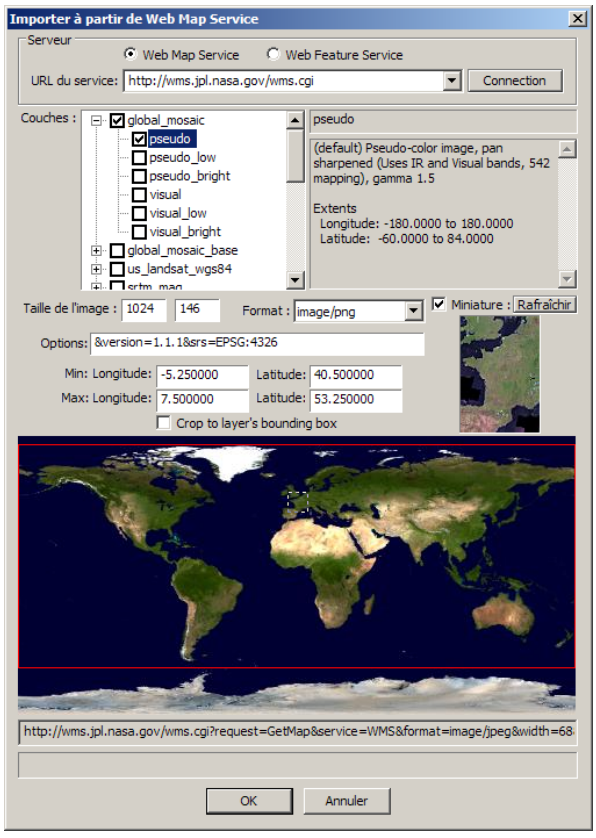


FIG. 5.3: *Import de données via un* Web Map Service *dans* `AutoDEM`.

`AutoDEM` permet également d'importer de l'information via les services Internet *Web Feature Service* (WFS) de l'OGC (service Web similaire au WMS mais prenant en charge des données vecteurs), TerraServer-USA et MapPoint de Microsoft.

Outils Un outil est une fonctionnalité permettant à l'utilisateur de travailler directement sur une couche du projet à l'aide de la souris et parfois du clavier. Les outils se distinguent par une icône particulière et sont regroupés au sein de boites à outils qui sont généralement spécifiques à une couche donnée. Il existe par exemple une boite regroupant les outils relatifs à la manipulation des courbes de niveau et permettant de créer, de supprimer, le déplacer, etc. des courbes.

Le développeur peut ajouter un nouvel outil dans l'interface en écrivant une classe C++ et en la déclarant à l'aide d'une fonction du SDK.

Filtres Un filtre est une fonction qui applique un traitement local ou global sur une ou plusieurs couches de données. A l'aide d'une chaîne de formatage, le développeur décrit les types des paramètres, leur valeur par défaut et leur intervalle de validité que l'utilisateur sera invité à saisir lors de l'exécution du filtre. `AutoDEM` génère ensuite automatiquement la fenêtre permettant à l'utilisateur de choisir les paramètres et lui offrant une zone de prévision (voir saisie d'écran 5.4).

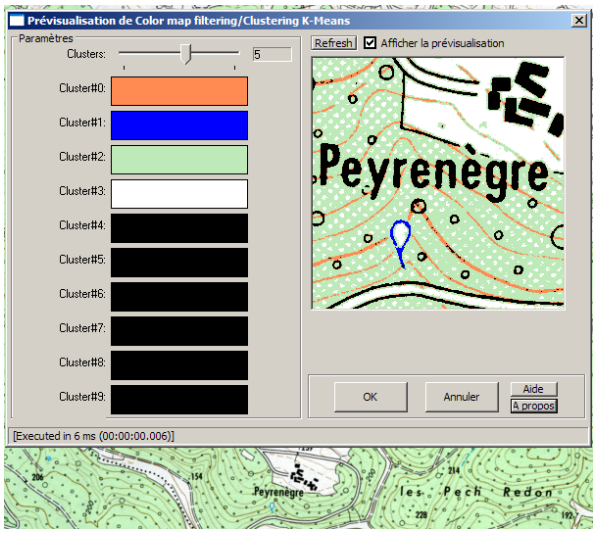


FIG. 5.4: *Paramétrage d'un filtre sous `AutoDEM`.*

Une fonction permet également au développeur de fournir une documentation spécifique pour chaque filtre.

Afin d'offrir à l'utilisateur un état d'avancement du traitement effectué par le filtre, le SDK fournit un ensemble de fonctions et macros pouvant être insérées pour indiquer à l'utilisateur la partie du traitement en cours et le temps restant pour traiter la tache à l'aide d'une barre de progression.

5.3 Applications

En marge des travaux présentés dans les précédents chapitres concernant la reconstruction de MNT à partir de cartes topographiques, nous avons développé, à l'aide d'`AutoDEM`, de nombreux autres outils d'analyse de cartes et de MNT.

5.3.1 Extraction de toponymes

L'extraction automatique de la couche toponymique est une tâche à part entière dans le cadre de l'extraction automatisée d'informations à partir de cartes papier numérisées. Cette tâche s'inscrit dans la thématique plus générale de la reconnaissance de caractères qui a suscité une très grande quantité de travaux scientifiques. Parmi ces travaux, quelques chercheurs [PD94][Vel02] se sont particulièrement intéressés au problème de la localisation et de l'identification des caractères alphanumériques situés sur des cartes numérisées. En vue d'une application de toponymes sur un terrain 3D, nous avons proposé [PGPG07] à ICDAR 2007, une technique heuristique divisée en 5 étapes et illustrée par le diagramme de la figure 5.5 : i) segmentation de la carte et extraction des objets de couleur noire (couleur la plus fréquente des toponymes) ; ii) analyse des composantes connexes et détection des composantes susceptibles d'être des caractères ; iii) création de chaînes de composantes connexes susceptibles de former des mots ; iv) image binaire formée par chacune des chaînes est analysée par un logiciel d'OCR ; v) post-traitement des chaînes reconnues et élimination des chaînes incohérentes.
La figure 5.6 montre un exemple de résultat obtenu sur un extrait de carte IGN. Une vue 3D du résultat est illustrée par la figure 5.7.

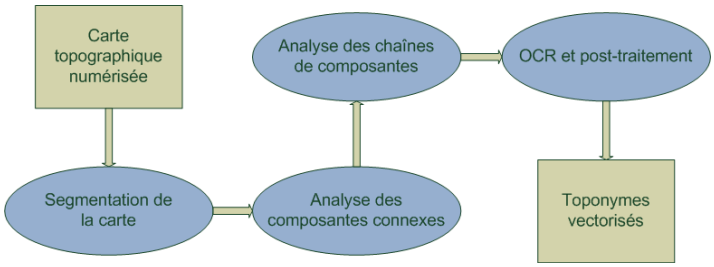
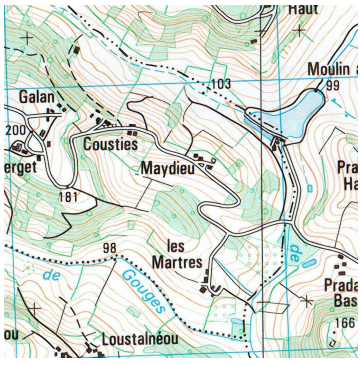
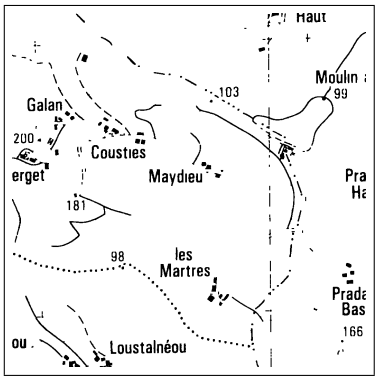


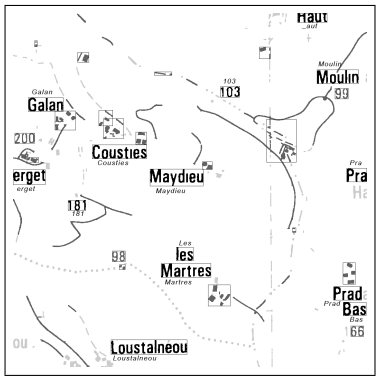
FIG. 5.5: *Diagramme de notre algorithme d'extraction de toponymes.*



(a) Extrait d'une carte topographique IGN.



(b) Segmentation de la carte par binarisation (seuil 128) suivi d'une fermeture morphologique.



(c) Les chaînes de composantes connexes restantes après l'analyse sont encadrées. Les mots en italique correspondent aux chaînes reconnues par l'OCR et corrigées par post-traitement.

FIG. 5.6: *Extraction de toponymes sur une carte topographique.*

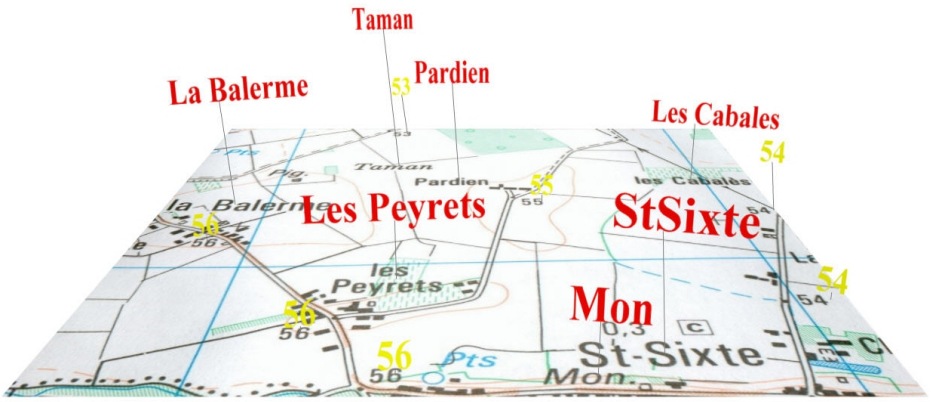


FIG. 5.7: *Exemple de visualisation des toponymes extraits en 3D dans une scène VRML97.*

5.3.2 Extraction de bâtiments

Nous nous sommes également intéressés à la reconstruction automatique de scènes urbaines 3D à partir de l'analyse de plans de masse de type cadastre. Après segmentation et vectorisation des différents segments, les différentes composantes connexes de chaque bâtiment sont utilisées pour reconstruire un modèle 3D en langage VRML97. En particulier, les arêtes des toits sont formées en utilisant le squelette de la surface du bâtiment. La figure 5.8 illustre la technique mise en place.

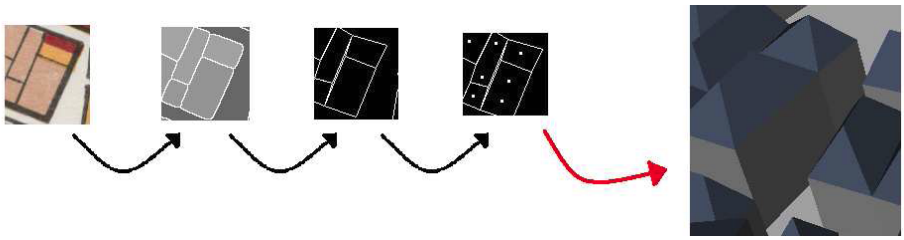


FIG. 5.8: *Chaine de traitement de la vectorisation d'empruntes de bâtiments.*

5.3.3 Analyse de MNA

L'analyse morphométrique d'un MNA cherche à détecter ou à quantifier des éléments caractéristiques de la surface. Grâce à la plate-forme d'`AutoDEM`, nous avons mis au point différents algorithmes tels que ceux décrits par Wood [Woo96]. Des représentations paramétriques locales du modèle sont utilisées pour caractériser la morphométrie de chaque point du modèle selon 6 classes : sommet, arête, plateau, puits, vallée ou col.

Nous avons également implémenté des techniques de détection des lignes de partage des eaux et des bassins versants inspirés par les travaux de Soille [Soi92] ainsi que des techniques de détection de champ d'intervisibilité, ou d'extraction des réseaux de drainage.

5.4 Utilisateurs

`AutoDEM` est actuellement distribué sous forme de *freeware*. Seuls les binaires sont disponibles au téléchargement, principalement sur le site dédié, mais également sur la plate-forme Tucows. Depuis la mise en ligne en 2004, plus de 700 téléchargements ont été recensés depuis de nombreux pays du monde.

Sur le site officiel, les personnes désirant obtenir le logiciel sont invitées à remplir un petit questionnaire. Grâce aux résultats de cette enquête, nous pouvons constater que les personnes intéressées se divisent en 4 classes :

- particuliers : cartographie pour jeux vidéo, applications GPS ;
- étudiants et chercheurs : disciplines informatique, archéologie, géographie, architecture, travaux agricoles ;
- personnels instituts publics d'étude du territoire, d'urbanisme ;
- personnels d'entreprises privées : agences d'urbanisme, cabinets d'architectes, de paysagisme, entreprises de jeux vidéo.

Parmi les différents utilisateurs, nous avons en été en relation avec l'Institut Ausonius (CNRS / Université Bordeaux 3). Spécialisés dans la reconstitution 3D de sites archéologiques, notre logiciel leur permet de reconstituer la topographie d'un lieu à partir des cartes topographiques récentes. La figure 5.9 présente la reconstruction du MNT autour du site de Chazal en Dordogne à partir d'une carte topographique de l'IGN en Dordogne. La reconstruction de la topographie d'un site turc ancien à partir d'une carte ancienne est illustrée dans la figure 5.10.

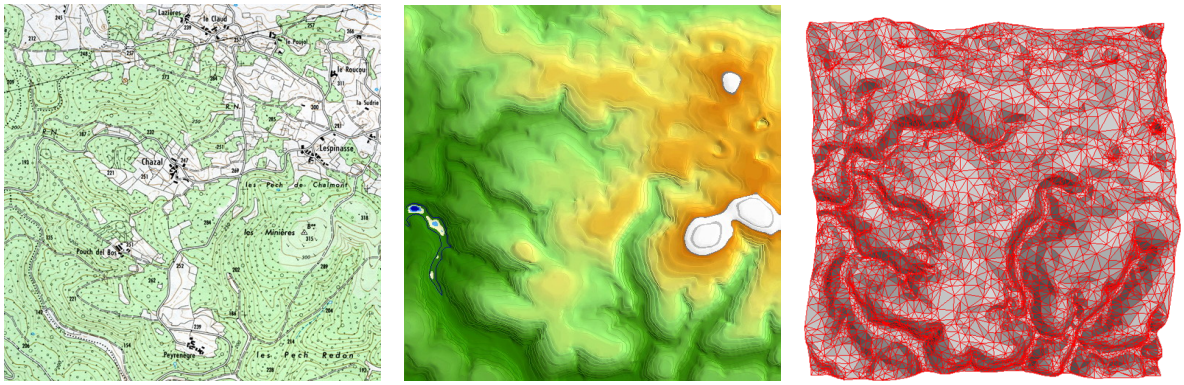


FIG. 5.9: *Création d'un MNA et d'un RTI autour du site de Chazal en Dordogne avec* `AutoDEM` *à partir d'une carte topographique IGN.*

5.5 Développements futurs

Comme tout logiciel, `AutoDEM` est amené à évoluer. Dans la mesure où, comme nous l'avons vu dans les chapitres précédents, la reconstruction de MNT à partir de cartes topographiques nécessite de nombreuses interactions avec l'opérateur, nous souhaitons apporter de nouvelles fonctionnalités à l'interface et surtout en simplifier l'usage. Par exemple, nous souhaitons créer, pour les différentes familles de tâches réalisables avec le logiciel, des assistants pour guider l'utilisateur dans les différentes étapes de traitement.

Une autre amélioration à venir porte sur la taille des documents traités. En effet, ac-

FIG. 5.10: *Création du MNA du site ancien d'Alazeytin Kalesi (Turquie)* `AutoDEM` *à partir d'une carte topographique ancienne.*

tuellement les couches rasters sont gérées comme des tableaux unidimensionnels alloués classiquement comme des zones contiguës en mémoire. Or les cartes topographiques sont des documents de grande taille, et numérisées avec une résolution suffisante ils peuvent nécessiter une grande quantité de mémoire. Il n'est pas rare d'avoir à traiter des images de tailles 10000×10000 voire plus et compte tenu du nombre de couches gérées par le logiciel, une saturation de la mémoire RAM et virtuelle peut survenir rapidement. Nous souhaitons donc intégrer à `AutoDEM`, un système de multi-résolution et de pavage de l'image pour permettre de visualiser efficacement les très grandes images. Pour le traitement de ces images, il sera également nécessaire de réfléchir à des techniques spécifiques d'accès rapides aux données et de parallélisation des traitements telles que celles utilisées par VIPS [CM96].

Enfin, à ce jour, il manque encore à notre logiciel une véritable documentation et celle-ci devra voir le jour rapidement afin de pouvoir toucher un plus grand nombre d'utilisateurs, en particulier ceux n'ayant pas connaissances préalables sur la chaîne de traitements à appliquer. Il faudra également arrêter une licence d'utilisation plus adaptée aux différentes utilisations possibles du logiciel en distinguant l'usage à fin de recherche de l'usage professionnel.

Deuxième partie

Visualisation interactive de vastes modèles de terrains

Les terrains numériques sont utilisés dans de nombreuses applications informatiques telles que celles reposant sur les SIG, mais aussi les jeux vidéos ou les simulateurs de vols. Dans un souci de réalisme toujours accru, il est nécessaire d'utiliser des modèles de terrains toujours plus grands et plus précis.

Cependant, visualiser de vastes terrains implique la manipulation de grands ensembles de données pouvant contenir des dizaines voir des centaines de millions d'échantillons, que ce soit des points, des triangles ou des *voxels* (éléments volumiques). Une telle complexité introduit deux grandes contraintes : i) la quantité de données impliquées peut être supérieure à la mémoire RAM disponible sur la machine hôte ; ii) le nombre de primitives graphiques à dessiner peut être trop grand pour se faire en temps réel sur une machine donnée.

Le niveau de détail

La technique du niveau de détail (NDD, en anglais *Level of detail*, LOD) consiste à adapter localement ou globalement le niveau de complexité (en terme de nombres de primitives 3D) utilisé pour afficher un objet. Les niveaux de détails sont le plus souvent choisis en fonction de la distance du modèle au point de vue. Un objet proche sera affiché plus grand qu'un objet lointain, un plus grand détail sera donc nécessaire pour afficher l'objet proche tandis que l'objet loin pourra se contenter d'être décrit grossièrement (voir l'illustration figure 5.11). En pratique, les différents niveaux de détail peuvent être modélisés manuellement, précalculés ou s'effectuer à la volée. La sélection du niveau de détail se fait elle en temps réel. Nous renvoyons le lecteur à l'ouvrage [LWC+02] pour une bonne introduction aux techniques de NDD.

De par leur taille, l'affichage de terrains est particulièrement concerné par le NDD. Les propriétés de ces terrains (géométrie plus contrainte généralement en 2.5D, grilles uniformément

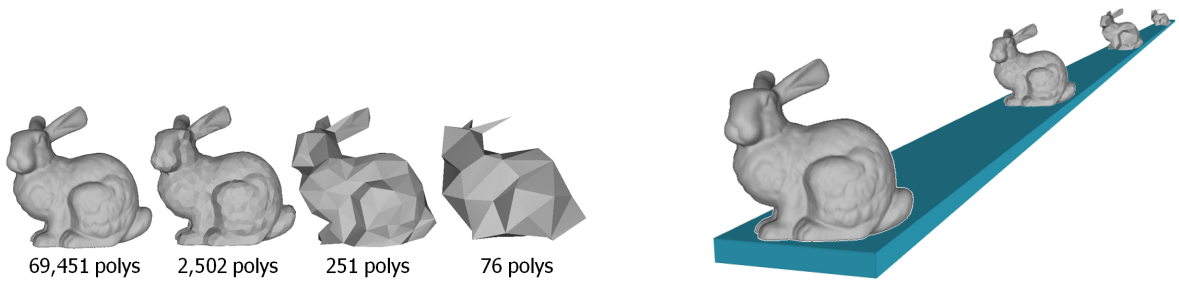


Fig. 5.11: *A gauche : différents niveaux de détail pour représenter un lapin. A droite : sélection du niveau de détail en fonction de la distance au point de vue.*

échantillonnées, MNA) requièrent des techniques spécifiques pour les traiter. Ces techniques sont confrontées aux deux difficultés suivantes : i) les modèles de terrains peuvent être extrêmement grands ; ii) le modèle est simultanément très proche et très éloigné du point de vue. Comme nous le verrons plus bas, des solutions sont entièrement basées sur des calculs effectués sur CPU alors que d'autres utilisent à la fois le CPU et le GPU (généralement à l'aide de programmes *shaders*). Parmi ces solutions, certaines nécessitent que l'ensemble des données soient disponibles en mémoire alors que d'autres sont dites "hors mémoire" (en anglais *out of core*) ou par téléchargement progressif.

Le téléchargement progressif

Le *streaming*, généralement traduit en français par diffusion en mode continu ou, plus techniquement, téléchargement progressif, est un principe très utilisé sur Internet. Cette technique permet de lire un fichier multimédia au fur et à mesure de son téléchargement. Le *streaming* s'oppose donc à la technique de téléchargement "classique" consistant à rapatrier l'intégralité des données d'un fichier avant d'en effectuer la lecture. Avec le développement d'Internet, l'accessibilité en haut-débit et l'apparition de nouveaux services de diffusion audio et télévisuels, les techniques de *streaming* connaissent actuellement une grande fortune, l'utilisateur pouvant en débuter l'écoute immédiatement.

Dans un contexte de mobilité où la qualité de la connexion est difficilement prévisible, le *streaming* de terrains est une technique extrêmement précieuse car elle permet à l'utilisateur une visualisation et une navigation immédiate, les données et donc le terrain visualisé gagnant en étendue et en précision au fil du temps.

Méthodes proposées

Dans cette partie, nous présentons deux grandes familles de techniques pour visualiser de vastes terrains sur des TMC. A partir des données stockées sur un serveur distant, la première famille transmet progressivement les données nécessaires au client TMC pour qu'il effectue lui même le rendu 3D en temps réel. Ces techniques sont possibles si le TMC dispose des capacités suffisantes pour réaliser ce rendu. Si tel n'est pas le cas, la seconde famille déporte le calcul du rendu de la scène 3D à un serveur dédié, l'image ou le flux d'images obtenu étant alors envoyés au TMC en temps réel.

Dans le chapitre 6 nous présentons une méthode permettant de naviguer en temps réel dans des scènes 3D décrivant de grands modèles de terrains texturés stockés sur un serveur local ou distant. Basée sur une transmission progressive et une gestion efficace de niveaux de détails, l'atout majeur de notre technique est son *adaptativité* aux ressources du client. Nous avons pu valider cette méthode de rendu sur différentes plates-formes allant d'un TM à une grappe de PC reliés à des vidéos-projecteurs.

Nous avons également mené des recherches sur des techniques dites de rendu à distance. Il ne s'agit plus alors d'effectuer le rendu en temps réel de gros modèles de terrains sur le client, mais de déléguer ces coûteux calculs à des serveurs dédiés. La première technique présentée dans le chapitre 7 repose sur une connexion haut débit entre le client disposant de faibles capacités de calculs et le serveur disposant de la base de données et d'un moteur de rendu. La seconde technique est un service ponctuel permettant d'offrir un panorama virtuel dans le cadre d'une application de repérage mobile en pleine nature.

Chapitre 6

Visualisation de larges terrains sur plates-formes hétérogènes

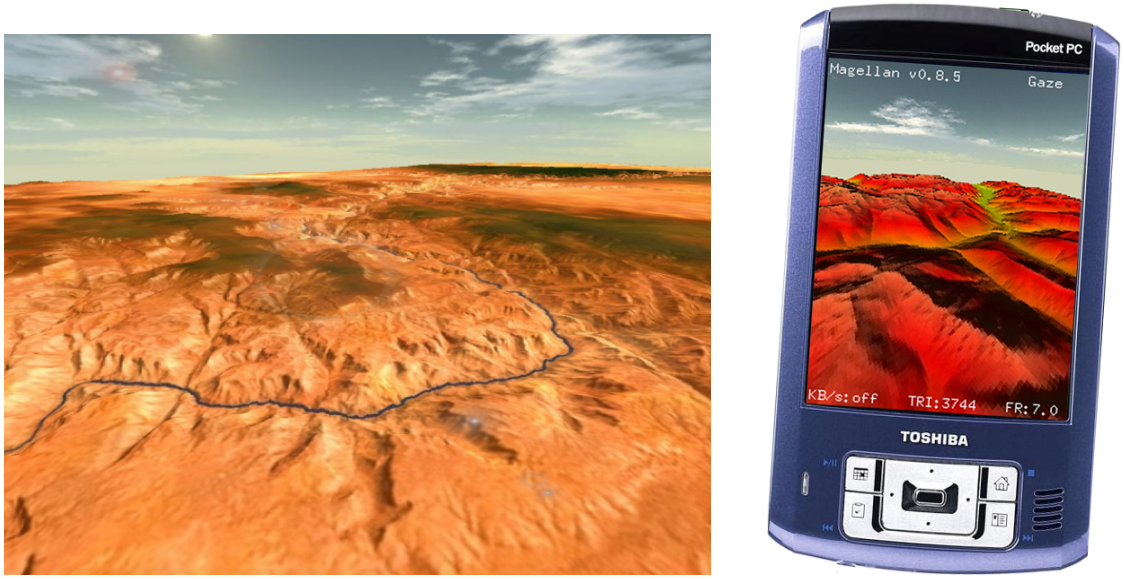


Fig. 6.1: *Téléchargement progressif et rendu adaptatif de grands terrains. A gauche : rendu adaptatif du modèle du Grand Canyon sur une machine de bureau effectué à 25tps, en affichant 440K triangles. A droite : rendu adaptatif du modèle du Puget Sound (situé dans l'état de Washington aux Etats-Unis) sur un PocketPC avec un taux de rafraîchissement fixé à 7tps en affichant 3744 triangles et téléchargé via une connexion USB 2.0.*

6.1 Introduction

La solution que nous proposons cherche à couvrir une large variété de configurations (voir figure 6.1). Aussi, nous avons choisi une approche de type *streaming* afin d'offrir une solution pouvant être distribuée sur un réseau, mais pouvant fonctionner tout aussi bien une sur architecture purement locale. A cet effet, nous utilisons un *pavage régulier* du modèle de terrain afin d'en effectuer son chargement progressif et de pouvoir l'adapter à la mémoire dispo-

nible sur la machine cliente. Les configurations réseaux visées impliquent de réaliser un rendu interactif sur différents types de terminaux, allant de la station de travail haut de gamme aux terminaux mobiles à faibles capacités. Afin de résoudre cette double exigence apparemment contradictoire, nous proposons une représentation multi-résolution originale, que nous avons baptisée *strip masks*, pour réaliser un rendu adaptatif de chaque tuile visible. Cette représentation, utilisée de concert avec une métrique d'importance visuelle et un allocateur de budget en polygones adaptatif, permet d'effectuer un rendu adaptatif côté client. Notre solution, grâce à ces aspects progressifs et adaptatifs, est donc largement portable et a été testée avec succès sur différentes configurations.

Notre solution peut être décomposée en deux parties. La première consiste en un téléchargement progressif de tuiles de MNT et à leur gestion côté client. Cette partie consiste à assurer l'existence de la plus grande région possible en fonction de la mémoire disponible autour du point de vue actuel de l'utilisateur. La seconde partie est dédiée au rendu adaptatif de cette zone en temps réel. Le but étant alors d'afficher le plus grand nombre de triangles à l'aide de la meilleure qualité de texture disponible tout en satisfaisant un taux de rafraîchissement fixé.

La technique décrite ici et développée en collaboration avec Jean-Eudes Marvie, a fait l'objet d'une publication dans une conférence internationale [PM05] en 2005.

La suite de ce chapitre est organisée de la façon suivante : après avoir rapporté les approches existantes dans la section 6.2, nous présentons au lecteur la plate-forme Elkano sur laquelle repose notre technique. Nous décrivons notre technique de gestion de tuiles adaptative dans la section 6.4 puis nous présentons notre technique ainsi que notre solution de rendu adaptatif dans la section 6.5. Enfin, les résultats expérimentaux obtenus sont décrits dans la section 4.3.4.

6.2 Revue des approches existantes

De nombreux travaux de recherche et d'ingénierie ont été menés durant la dernière décennie dans le domaine du rendu interactif de terrain. Dans cette section, nous distinguons deux grandes familles de méthodes, même si, en pratique, il existe parfois des passerelles permettant de passer de la première à la deuxième.
La première regroupe les techniques prenant en charge les modèles tenant entièrement sur la mémoire de la machine hôte. La seconde famille, dite de techniques hors mémoire, rassemble les algorithmes spécifiquement crées pour visualiser les très gros terrains dont la quantité de données nécessaire ne peut pas être chargée entièrement en mémoire. C'est dans ce deuxième cadre que notre solution se situe plus particulièrement, mais l'étude des premières se révèle fort instructive.

6.2.1 Techniques en mémoire

La plupart des techniques décrites ici sont basées sur la gestion d'un maillage triangulé irrégulier. Le maillage est raffiné en temps réel selon différentes stratégies. En 1996, Lindstrom [LKR+96] introduit une technique permettant de changer de niveau de détails de façon continue (CLOD, *Continuous LOD*). Cette technique repose sur un maillage défini par des blocs de triangles droits subdivisés récursivement selon une métrique basée sur l'erreur du maillage projeté.Deux ans plus tard, [RHSS98] propose des améliorations à la technique de Lindstrom. Il intègre d'une part une solution dite de *geomorphing* des vertex afin de réduire

l'effet de saut (*popping*) produit lors du changement de niveau. D'autre part, il y adjoint une métrique basée sur la distance du point de vue et la rugosité du terrain. Dans [Hop96], Hoppe propose une technique de maillage progressif (*progressive meshes*) dépendant du point de vue et décrit plus tard son application dans le cadre du rendu de terrains [Hop98]. Dans [DWS+97], Duchaineau *et al.* décrivent leur technique de *ROAMing*, une méthode très efficace basée sur des diamants de triangles gérés par des opérations de division-fusion (*split and merge*) à l'aide de files de priorités (voir figure 6.2). Même si cet algorithme a été beaucoup utilisé dans l'industrie du jeu vidéo, son implémentation est délicate selon [Blo00]. Plus récemment, [Lev02] proposèrent de réduire le coût CPU des méthodes précédentes basées sur des arbres binaires de triangles en manipulant des agrégats de triangles au lieu de triangles simples. Comme ces agrégats de triangles sont utilisés dans plus d'une trame, ils peuvent être mis en cache dans la mémoire de la carte graphique et ainsi engendrer une accélération significative.

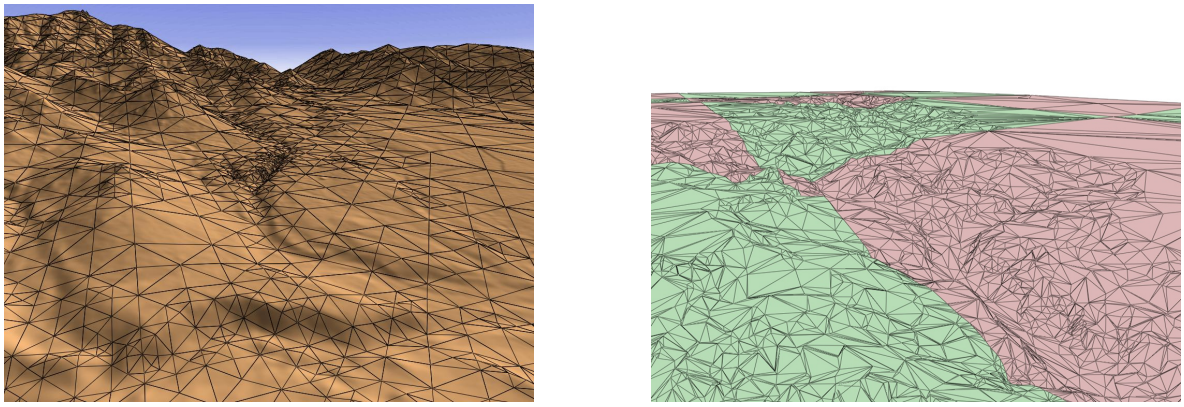


FIG. 6.2: *A gauche : Triangulation par CLOD de Lindstrom [LKR+96]. A droite : Maillage progressif de Hoppe [Hop98]*

Losasso*et al.* [LH04] expliquent que les algorithmes précédents induisent des accès aléatoires en mémoire et un mode de rendu immédiat fortement pénalisant. De plus, ils furent mis au point avant le développement des processeurs graphiques 3D et impliquent donc souvent des calculs CPU coûteux. De nos jours, les GPU sont capables d'afficher plusieurs millions de triangles par seconde et sont extrêmement efficaces dans le traitement de chaînes de triangles (*triangle strips*). En conséquence, il est désormais intéressant d'imaginer des algorithmes tirant avantage de ces capacités.

Dans un papier récent, Losasso et Hoppe [LH04] proposent d'appliquer le concept du *clipmapping* [TMJ98], permettant d'utiliser efficacement de très grandes textures dans le rendu temps réel de scènes 3D, au rendu de la géométrie de gros MNA. Leur méthode exploitant au maximum les capacités accélératrices des GPU se base sur un ensemble de grilles régulières imbriquées centrées autour du point de vue. La continuité géométrique est garantie par l'utilisation de régions de transition entre deux niveaux de grilles calculées à l'aide d'un *vertex shader*. Afin de pouvoir traiter l'affichage de modèles volumineux, les auteurs utilisent également un algorithme de compression efficace permettant de les charger entièrement en mémoire centrale. Bien qu'efficace, cette technique nécessite toujours des calculs CPU pour calculer les indices des vertex à afficher à chaque image. C'est pourquoi une extension a été proposée par Asirvatham et Hoppe [AH05] pour augmenter encore la quantité de calculs transférés du CPU au GPU. Cependant, même si cette méthode est très efficace sur des machines équipées

de processeurs graphiques 3D récents, elle souffre de reposer intensivement sur l'utilisation de programmes *shaders*, qui ne sont malheureusement pas disponibles sur des terminaux légers. Enfin, si cette technique s'applique bien dans le cas d'une visualisation de données disponibles localement, elle n'est pas praticable dans le cadre d'applications réparties, reposant sur des transferts réseau et des machines aux capacités variables tant au niveau des ressources de calcul que des ressources mémoires.

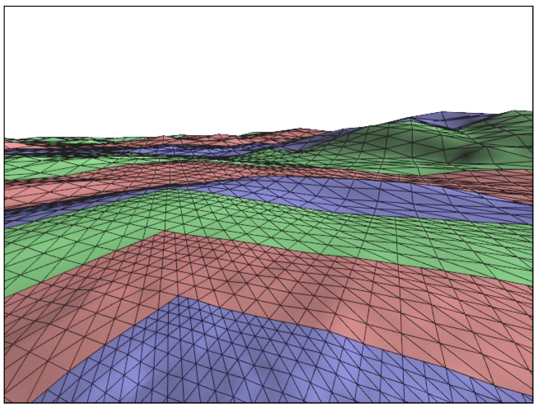


FIG. 6.3: *Imbrication de grilles régulières utilisée dans la technique du* GeoClipmap.

6.2.2 Techniques hors mémoire

Les techniques hors mémoire permettent d'afficher des terrains d'une taille extrêmement grande sans se soucier de la limitation de la mémoire RAM. Les données topographiques sont chargées lorsqu'elles deviennent nécessaires et libérées lorsque tel n'est plus le cas.

Dans [Paj98], Pajarola étend la triangulation par *quadtree* de Lindstrom [LKR+96] avec un algorithme différent de sélection des vertex et une méthode de construction des chaînes de triangles (*triangle strips*) efficace. La méthode est combinée avec un algorithme de gestion de la scène dynamique et progressif permettant un rendu hors mémoire. Plus récemment, Cignoni *et al.* [CGG+03b, CGG+03a] ont décrit une technique pour gérer et afficher de larges terrains texturés de façon hors mémoire baptisée *batched dynamic adaptive meshes* (BDAM). La technique du BDAM est basée sur un arbre binaire de petits morceaux de surface triangulée (TIN) qui sont calculés et optimisés hors ligne. La visualisation se fait ensuite à l'aide d'un modèle de communication optimisé entre le CPU et le GPU, l'exploitation du GPU programmable, une représentation compressée hors mémoire et une technique de préchargement spéculative pour atténuer la latence du disque dur. Ces solutions donnent de très bons résultats sur des machines puissantes, mais sont impraticables dans notre contexte de mobilité dans la mesure où les GPU ne sont pas encore programmables et où il est impossible d'assurer une latence faible entre la base de données distante et la mémoire graphique de l'ordinateur. En outre, ces solutions présentent des coûts CPU non négligeables.

Visant la distribution de contenu sur le Web, Reddy *et al.* [RLIB99] décrivent la solution *TerraVision II*, sorte d'ancêtre de Google Earth, un navigateur de terrains géoréférencés au format VRML97. Cette solution repose sur une pyramide de modèles de terrains qui est précalculée *off-line* (voir figure 6.4). L'approche consiste ensuite à utiliser le nœud `LOD` du VRML97 pour y décrire chaque niveau de hiérarchie avec des nœuds `ElevationGrid`. Cette

technique engendre donc une grande redondance des données et aucun soin n'est pris pour assurer la continuité entre les différents niveaux de détails des différentes grilles. L'aspect client/serveur est pris en compte implicitement par l'utilisation du VRML et les données peuvent être obtenues via des URL et donc un serveur HTTP.

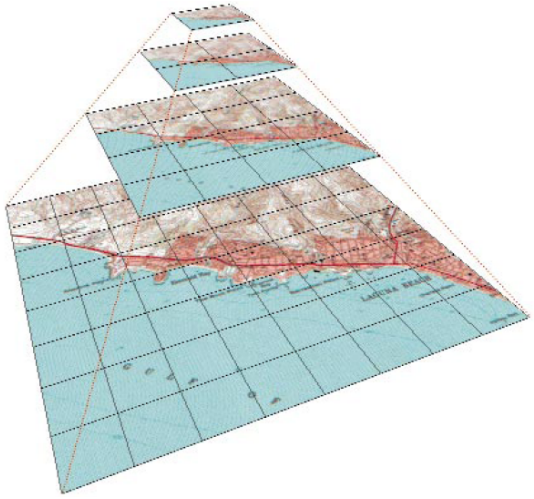


FIG. 6.4: *Pyramide utilisée par* TerraVision II *illustrant 4 niveaux de détails où chaque tuile fait 128×128 points.*

Une autre solution de téléchargement progressif proposée par Aubault [Aub03] repose sur un encodage de la topographie sous forme d'ondelettes et permet ainsi un téléchargement progressif des données et un rendu multi-résolution. Bien qu'efficace, cette solution nécessite des calculs coûteux côté client.

6.3 Plate-forme de développement

La technique que nous proposons est implémentée sur la plate-forme `Elkano`, elle même issue de la plate-forme `Magellan`.

6.3.1 `Magellan`

`Magellan` est une plate-forme de visualisation mise au point par J.-E. Marvie durant sa thèse [Mar04] à l'IRISA (Projet SIAMES) encadrée par K. Bouatouch. Cette plate-forme écrite en C++ a pour but de faciliter le développement de nouvelles solutions de visualisation de scènes 3D interactives distribuées sur des machines hétérogènes (voir figure 6.5).

Techniquement, la plate-forme `Magellan` offre :

- un ensemble de classes systèmes encapsulant des appels systèmes (*sockets*, *threads*, etc.) afin d'assurer la portabilité ;
- un méta-graphe de scène et des classes de nœuds pouvant être distribués ;
- un système de modules offrant la possibilité d'enrichir la boucle principale interaction-rendu ;
- un système de plug-ins permettant de décrire de nouveaux nœuds utilisables dans le graphe de scène ;
- le support du langage de description de scènes VRML97 ;

– un système de moniteurs de performances (par exemple analyse du taux de rafraîchissement ou taux d'occupation mémoire) et d'allocateurs de budgets permettant de simplifier les solutions adaptatives ;
– un ensemble d'applications génériques dont un serveur et différents clients.

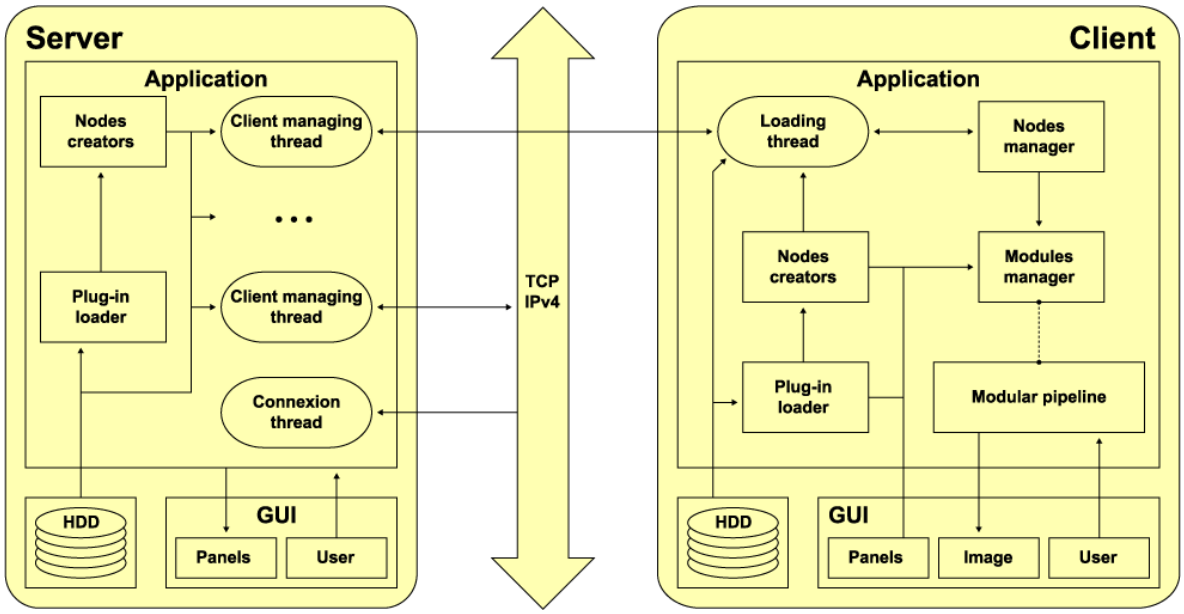


FIG. 6.5: *Schéma d'une application générique distribuée avec Elkano.*

Un avantage extrêmement important est l'aspect *cross-platform* (Windows / Linux / SunOS) du logiciel.

6.3.2 `Elkano`

Durant son post-doctorat au sein de l'équipe Iparla, nous avons, avec J.-E. Marvie, procédé au portage de la plate-forme `Magellan` sur les TMC basés sur le système d'exploitation Windows Mobile (cas des PocketPC et Smartphone en particulier). La nouvelle version de la plate-forme, baptisée `Elkano`, à été l'objet d'une nouvelle phase de maintenance et développements que nous avons menée et incluant :

– le support de la plate-forme Windows Mobile et donc la compatibilité avec les TMC ;
– un module de rendu distribué pour une grappe de PC reliés par un réseau TCP/IP : chaque machine se voit attribuée une partie de la zone d'affichage et une machine gère les interactions utilisateur et fait office de chef d'orchestre en assurant une barrière de synchronisation avant chaque passe de rendu. Cette technique simple mais efficace est en particulier utilisée pour effectuer du rendu temps réel sur la grande surface d'affichage de la salle de réalité virtuelle du LaBRI, Hémicyclia 2 ;
– la prise en charge des nœuds GeoVRML et d'un module d'interaction géoréférencé pour permettre la visualisation de scènes géoréférencées ;
– la prise en charge du langage de description de scènes X3D ;
– la capacité pour le serveur de construire un arbre des scènes disponibles dans sa base de données via un référencement par fichiers XML ainsi que la possibilité pour le client d'accéder à cette liste afin de simplifier la sélection de la scène pour l'utilisateur.

Ce portage fut pour nous l'occasion de développer `GLUT|ES`, une version adaptée à Windows Mobile et à OpenGL—ES (voir Annexes).

6.4 Gestion adaptative du pavage

Afin de réaliser la transmission et la gestion du terrain, nous utilisons le système pagination assez similaire aux techniques décrites dans [Paj98, RLIB99, ZZSP01, LC03]. La base de tuiles régulières est réalisée en pré-traitement côté serveur par subdivision du MNA et de sa texture. La géométrie de chaque tuile est codée dans un fichier VRML tandis que la texture est codée dans un fichier JPEG ou dans un format de texture progressive. L'encodage des tuiles en fichiers séparés permet un téléchargement (à l'aide d'un simple protocole de transfert de fichier) et une gestion à l'aide d'un simple tableau 2D. Un fichier principal contient la description des tuiles du modèle (taille des tuiles, nombre de tuiles, position, géoréférencement, etc.) est initialement téléchargé. Ces informations sont ensuite utilisées pour gérer le pavage adaptatif.

L'algorithme de gestion des tuiles que nous avons mis au point a pour but de maintenir la plus grande zone carrée, faite de bandes carrées de tuiles appelées *ceintures*, autour du point de vue. Ce carré de tuiles assure à l'utilisateur la possibilité de tourner sur lui même et de visualiser immédiatement le terrain sans avoir à attendre le téléchargement de nouvelles tuiles. Cependant, la taille de cette zone est fonction de la mémoire RAM disponible, la rendant adaptative à la machine utilisée pour la visualisation. La quantité de mémoire à utiliser pour le stockage des tuiles peut aussi être modifiée à chaud par un paramètre utilisateur. Connaissant la quantité de mémoire pouvant être utilisée, la zone visible est maintenue en suivant la position du point de vue et en téléchargeant de nouvelles tuiles ou en supprimant des tuiles de la mémoire.

L'algorithme 4 est exécuté avant la génération de chaque nouvelle trame. L'indice d'une ceinture représente le plus petit nombre de tuiles qui sépare la tuile courante (qui contient le point de vue) de la ceinture en question. Par exemple, dans la figure 6.6a, l'indice de la ceinture complète (dont toutes les tuiles ont été téléchargées) la plus grande est 2. La procédure de téléchargement est asynchrone et est réalisée en parallèle de la phase de rendu. Le téléchargement est toujours effectué par ceinture complète et démarré uniquement si la dernière requête de ceinture a été totalement remplie.

Si le point de vue ne bouge pas, l'algorithme téléchargera toutes les ceintures de tuiles pouvant tenir dans la taille mémoire impartie (voir figure 6.6a). Si le point de vue se déplace sur une tuile adjacente à la précédente, l'algorithme va tenter de maintenir le carré de ceintures autour de celle-ci (qui devient la tuile courante) en téléchargeant les tuiles manquantes, comme illustré dans la figure 6.6b. Dans cet exemple, toute la mémoire allouée a été consommée au pas décrit dans la figure 6.6a. L'algorithme va donc supprimer les tuiles lointaines afin de libérer de la mémoire pour permettre le téléchargement des nouvelles tuiles. A noter que l'algorithme prend également en compte implicitement le cas où le point de vue sauterait sur une tuile non adjacente à la précédente.

Dans d'autres cas, la mémoire peut ne pas être saturée lors du changement de tuile courante, par exemple lorsque le point de vue se déplace avant la saturation. Les tuiles restantes, ne formant pas une ceinture complète, sont alors conservées dans un cache pour une utilisation ultérieure éventuelle. La figure 6.6c illustre cette mise en cache. Pour réaliser cette copie d'écran, nous avons d'abord attendu la saturation à partir d'un bord du terrain (point A)

if aucun téléchargement en cours **then**
  i_c = indice de la plus grande ceinture complète
  if mémoire utilisée > limite mémoire **then**
    i_p = indice de la ceinture (parfois incomplète) la plus lointaine
    if $i_p > i_c$ **then**
      supprimer toutes les tuiles de la ceinture partielle de la mémoire
    end if
  else
    demander le téléchargement des tuiles manquantes de la ceinture $i_c + 1$
  end if
else
  initialiser chaque tuile nouvellement reçue
end if

Algorithme 4 : Gestion adaptative et asynchrone des tuiles.

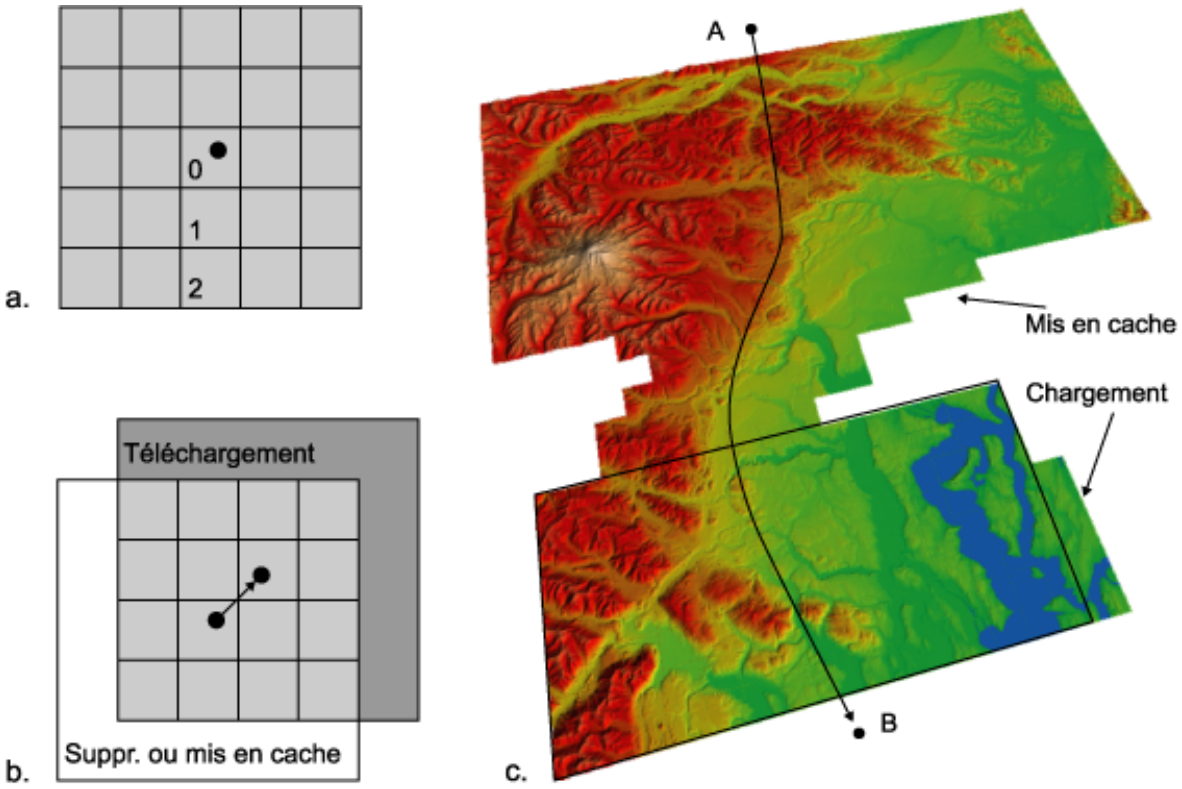


FIG. 6.6: *Gestion des tuiles et mise en cache. a) Zone carrée faite de 3 ceintures de tuiles centrées autour du point de vue. b) Préservation de la zone carrée lors du déplacement du point de vue. c) Illustration de la mise en cache : toutes les tuiles disponibles en mémoire sont affichées. Seule la zone rectangulaire encadrée en noir est normalement rendue.*

avant d'effectuer un déplacement rapide vers le bord opposé (point B). On peut distinguer sur cette image les tuiles mises en cache ainsi que les tuiles téléchargées pour construire des ceintures complètes afin d'élargir la zone de terrain rendue.

L'algorithme 4 prend également en compte le fait que, même lorsque le budget mémoire est atteint, une ceinture complète n'est pas supprimée si elle contribue à la zone carrée visible. Cette contrainte assure la stabilité de l'algorithme adaptatif. Cependant, le budget mémoire peut parfois être légèrement dépassé.

Enfin, différentes stratégies peuvent être utilisées dans le cas où le point de vue sortirait de la zone de définition du terrain. La première pourrait bien entendu être de restreindre les déplacements de l'utilisateur sur la zone de terrain disponible. La seconde, que nous avons préférée, consiste à laisser à l'utilisateur la possibilité d'en sortir, la tuile courante utilisée par l'algorithme étant alors la tuile la plus proche du point de vue. Il faut noter que dans ce cas là, la zone carrée devient une zone rectangulaire.

6.5 Rendu adaptatif

Dans cette section nous présentons l'étape de rendu des tuiles chargées qui sont visibles depuis le point de vue. La sélection des tuiles visibles est effectuée classiquement, à l'air d'un *frustum culling* sur les boîtes englobantes des tuiles.
Afin d'afficher de façon adaptative les tuiles visibles, nous utilisons une structure de donnée multi-résolution appelée *strip masks* (section 6.5.1). L'étape de rendu est ensuite réalisée comme suit. Nous calculons d'abord l'importance visuelle de chaque tuile en utilisant la rugosité de la tuile et sa distance du point de vue (section 6.5.2). Ces importances visuelles sont ensuite utilisées pour partager un budget global de triangles, prédit pour assurer un taux de rafraîchissement donné, entre toutes les tuiles visibles. Pour chaque tuile, le budget partiel est finalement utilisé pour choisir le *strip mask* respectant le budget pour afficher la tuile (section 6.5.3).

6.5.1 Structure de données partagée : le *strip mask*

Une tuile de terrain est un MNA de résolution $(l \times h)$ décrivant l'altitude des points du terrain échantillonné. Notre implémentation prend en charge des tuiles de tailles quelconques, mais par souci de simplicité, nous considérerons ici le cas spécifique de tuiles de tailles $l = h = (2^n + 1)$. La représentation mémoire d'une tuile est un tampon de structures de vertex qui stockent les coordonnées 3D ainsi que les propriétés associées telles que les coordonnées de texture, la couleur et la normale.

La structure multi-résolution que nous proposons consiste en une pile de *strip masks* de différentes résolutions (voir figure 6.7). Un *strip mask* décrit une triangulation de la surface d'une tuile à une résolution donnée. Dans la pratique, un tel masque décrit une chaîne de triangles OpenGL à l'aide d'un tampon d'indices énumérant les vertex à utiliser pour construire le maillage relativement au tampon de vertex. L'utilisation de chaînes de triangles permet de tirer parti des optimisations de rastérisation de ces chaînes de triangles dans les cartes graphiques. De plus, un même masque pouvant servir durant plusieurs trames, nous pouvons également tirer parti des *listes d'affichages* ou de la technologie des *buffer objects* d'OpenGL.

L'avantage principal de cette structure de données tient au fait que toutes les tuiles de même résolution peuvent la partager. Dans la plupart des cas, une seule pile de masques peut donc être utilisée pour modéliser et rendre toutes les tuiles du terrain. En effet, chaque tampon d'indices de la pile de masques peut être utilisé pour n'importe quel tampon de vertex qui décrit les différentes tuiles du terrain. Cette unicité permet de diminuer le temps de calcul et surtout la quantité de mémoire utilisée. De plus, le calcul des masques est fait de façon

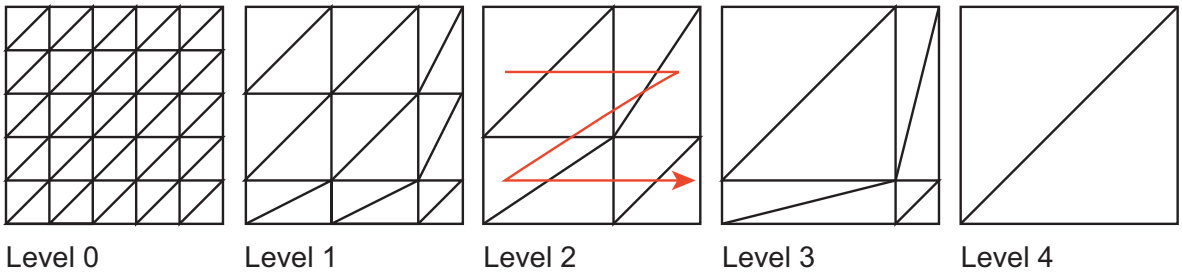


FIG. 6.7: *Pile de masques pour une grille de taille* 6×6. *La flèche indique l'ordre de description des triangles dans la chaîne de triangles. A noter que les niveaux 2 et 3 contiennent tous les deux 8 triangles. Dans ce cas, le niveau 2 sera choisi pour un budget entre 8 et 17 triangles. Le niveau 3 pourra être utilisé lors d'une transition entre les niveaux 2 et 4 comme expliqué dans la section 6.5.3.*

paresseuse, la première fois qu'ils sont nécessaires afin de distribuer le temps de calcul dans le temps lors des premières phases de rendu.

Nous définissons un masque de niveau n comme le réseau de triangles connectant les vertex de coordonnées (i, j) (dans la grille de taille $l \times h$) dont les indices sont congrues modulo $n + 1$. La pile de masques contient donc $\max(l, h) - 1$ masques. Un exemple d'une telle pile est illustré dans la figure 6.7. Cette approche est très différente des approches précédentes qui cherchent à optimiser la triangulation localement en fonction de la propriété de la surface. Ces approches permettent de décimer un maillage plus fidèle à la topographie sous-jacente, mais consomment plus de ressources CPU. Or les cartes graphiques actuelles sont capables d'afficher toujours plus de triangles en parallèle au CPU. Aussi, nous pensons comme [LH04] qu'il est préférable de rendre des maillages de plus grandes résolutions plutôt que de passer du temps à l'optimiser localement. La capacité de calcul CPU préservée peut alors être utilisée à d'autres fins : simulation, interaction, etc. Notre structure de données permet de simplifier le rendu d'une tuile à la sélection du masque en fonction du budget en triangles β alloué à cette tuile. La sélection se fait en $O(1)$ simplement en choisissant le niveau n tel que $T(n) <= \beta < T(n + 1)$.

Les *strips masks* peuvent être stockés dans des listes d'affichages ou des *Element Buffer Object* (EBO) s'ils sont disponibles sur le client, et ce, afin de minimiser les transferts entre la mémoire centrale et la mémoire graphique.

6.5.2 Importance visuelle des tuiles

L'importance visuelle est un pourcentage attribué à chaque tuile visible en fonction de ses caractéristiques intrinsèques et de sa position par rapport au point de vue. L'idée de base est de donner une importance plus grande, et donc plus de détails géométriques, aux tuiles proches ou à celles représentant un dénivelé fort (montagnes) qu'aux tuiles lointaines ou plates. Comme nous le verrons dans la prochaine section, les importances visuelles sont ensuite exploitées pour partager le budget global de triangles entre chaque tuile visible permettant ensuite de choisir la résolution du masque à utiliser.

L'algorithme 5 décrit comment nous calculons les importances visuelles des tuiles.

Dans une première boucle, nous calculons pour chaque tuile t : la distance $dist_t$ correspondant à la distance entre le point de vue et le centre de la tuile t, et $hauteur_t$ la hauteur de

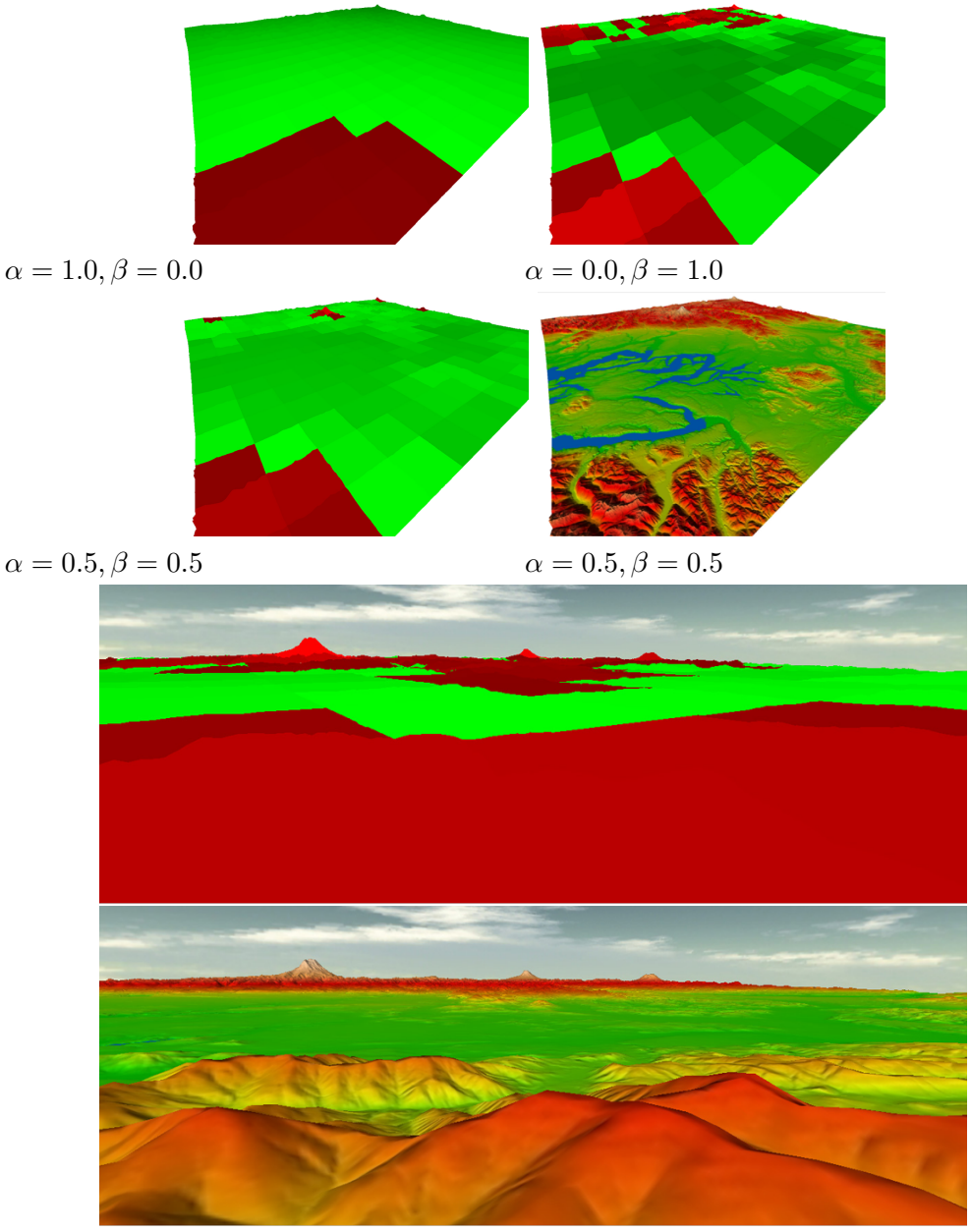


FIG. 6.8: *Importances visuelles du modèle du Puget Sound. Les altitudes du modèle sont exagérées afin de mieux percevoir le relief. La couleur associée à chaque tuile dépend de l'échelle suivante : le rouge est plus important que le vert, et une couleur claire l'est plus qu'une couleur sombre. En haut : de gauche à droite, les images montrent respectivement l'importance en utilisant la distance seulement, la hauteur seulement, les deux (avec* $\alpha = \beta = 0.5$*). La dernière image montre le terrain texturé. En bas : l'image de gauche montre l'importance visuelle en utilisant* $\alpha = \beta = 0.5$ *qui, expérimentalement, est un bon compromis. L'image de droite montre le résultat texturé. Notez que la forme des montagnes lointaines est préservée permettant d'obtenir un horizon crédible.*

Θ = ensemble des tuiles visibles dans le frustum
for chaque tuile t de Θ **do**
  $dist_t$ = distance entre la caméra et le centre de la tuile t
  $max\_dist = \max(dist_t, max\_dist)$
  $hauteur_t$ = dénivelé de la tuile t
  $sum\_dist = sum\_dist + dist_t$
  $sum\_hauteur = sum\_hauteur + hauteur_t$
end for
for chaque tuile t de Θ **do**
  calculer imp_t avec l'équation (6.1)
end for

Algorithme 5 : Calcul des importances visuelles des tuiles.

la boîte englobante de la tuile t. A noter que cette hauteur pourrait être remplacée par une mesure topographique tel que l'indice de rugosité (*Terrain Ruggedness Index*) introduit par Riley [RDE99]. Durant cette boucle, nous stockons également max_dist, la distance maximum $dist_t$ parmi toutes les tuiles visibles qui ont été traitées et nous accumulons $dist_t$ et $hauteur_t$ dans sum_dist et $sum\_hauteur$ respectivement. Enfin, dans une seconde boucle chaque importance de tuile imp_t est calculée comme la somme pondérée de la distance normalisée des valeurs $dist_t$ et $hauteur_t$ comme suit, avec $\alpha + \beta = 1$:

$$imp_t = \alpha \times \frac{max\_dist - dist_t}{sum\_dist} + \beta \times \frac{hauteur_t}{sum\_hauteur} \tag{6.1}$$

Les poids α et β sont choisis empiriquement pour accentuer ou discriminer les facteurs de distance ou de hauteur.

Les valeurs d'importance visuelle imp_t obtenues sont ensuite normalisées de telle façon que $\sum_t imp_t = 1$. La figure 6.8 illustre les importances visuelles obtenues en utilisant différentes valeurs pour (α, β). Empiriquement, un compromis entre distance et hauteur avec $\alpha = \beta = 0.5$ a montré des résultats satisfaisants lorsque la direction de la caméra est horizontale en préservant un horizon suffisamment précis tout en assurant une bonne résolution sur les tuiles proches.

6.5.3 Sélection du masque et rendu

Une fois les valeurs d'importances visuelles normalisées imp_t calculées, chaque tuile est indexée avec sa valeur d'importance visuelle dans la table de rendu de la plate-forme Elkano. Le moteur de rendu utilise alors ces valeurs d'importance pour partager un budget global en triangles entre les différentes tuiles. Le budget global est déduit de l'analyse des performances obtenues lors des trames précédentes en utilisant le taux de rafraîchissement obtenu relativement au nombre de triangles dessinés. Le budget calculé permet donc de maintenir le taux de rafraîchissement désiré par l'application. Le partage du budget est réalisé à l'aide d'un algorithme glouton privilégiant les tuiles de grande importance. Concrètement, si le budget global pour la trame courante est τ triangles, la tuile t d'importance imp_t recevra un budget de $imp_t \times \tau$ triangles. Une fois le masque sélectionné par la tuile concernée, le nombre de triangles non utilisés est réintroduit dans le budget global et pour pouvoir être utilisé par les tuiles suivantes de moindre importance. Pour plus de détails sur l'allocateur de budget le lecteur peut se référer à [Mar04, chapitre 5].

Une fois le budget réparti, le moteur de rendu procède de façon itérative au rendu proprement dit de chaque tuile. Notre représentation ne permet pas un déplacement continu de chaque vertex entre les différents niveaux. Cependant, pour éviter des sauts trop brutaux entre deux niveaux très différents, nous effectuons un changement progressif du niveau n au niveau $n+i$, avec $\mid n \mid > 1$, par pas de 1. Cependant, dans les niveaux de résolutions les plus grossiers, les sauts restent perceptibles.

Chaque tuile est généralement texturée avec une texture 2D, une photo satellite par exemple. Dans notre implémentation, les textures sont gérées comme des textures VRML97 classiques enrichies d'un format de fichier gérant des textures progressives décrit dans [MB03]. Ce fichier encode de façon efficace les niveaux de mipmap de la texture et permet un transfert progressif et adaptatif de ceux-ci au client. Cette représentation multi-résolution est également utilisée dans la phase de rendu afin d'optimiser l'occupation de la mémoire graphique (GRAM) ainsi que les transferts sur le bus graphique (AGP ou PCIExpress par exemple). Le point important avec cette solution qui est un plug-in de rendu Elkano, est qu'elle bénéficie de l'API disponible pour gérer les importances visuelles, calculer les budgets ou mettre à jour les niveaux de mipmap. De plus, l'importance visuelle calculée pour chaque tuile est utilisée non seulement pour la géométrie, mais également pour déterminer la résolution de la texture à utiliser.

Enfin, pour donner un effet plus réaliste au terrain, il est parfois intéressant de simuler l'éclairage du terrain à l'aide d'une source lumineuse telle que le soleil. Cet éclairage requiert le calcul des normales sur les faces ou sur les vertex du modèle. Quand les conditions lumineuses sont supposées constantes, la stratégie la plus efficace consiste à précalculer l'éclairage du terrain et à le stocker dans la texture. Dans d'autres cas, les normales doivent être recalculées et stockées dans le tampon de vertex à chaque changement de masque.

6.5.4 Continuité de la surface

Quand les niveaux des masques sont très différents entre deux tuiles adjacentes, des *T-vertex* deviennent visibles à la frontière des tuiles et des cassures apparaissent à la surface du terrain, laissant apparaître l'arrière-plan de la scène. La figure 6.11a illustre ce déplaisant problème. Les approches classiques [LC03, LH04] pour résoudre ce problème consistent à modifier la géométrie des tuiles sur les bords en introduisant de nouveaux vertex et de nouvelles arêtes.

De telles techniques ne sont pas compatibles avec notre technique basée sur des chaînes de triangles pré-calculées afin de réduire la charge de calcul sur le CPU. De plus dans notre modèle, les tuiles sont d'une certaine manière autonomes ce qui signifient qu'elles ne connaissent pas le niveau de résolution de leurs tuiles adjacentes. Une autre technique couramment utilisée appelée *filleting*, utilisée par exemple par Sun[1] ou par le visualiseur terrestre NASA World Wind consiste à ajouter une bande verticale de triangles, une sorte de filet (voir figure 6.9, autour les frontières de chaque tuile. Cette bande s'étire sous la surface du terrain. Chaque côté du filet est texturé en étirant le *texel* du bord de la texture correspondante. Cette solution est rapide, mais l'effet obtenu n'est pas toujours satisfaisant, car l'effet d'étirement de la texture est souvent visible comme l'illustre la figure 6.10.

Pour palier à cet effet, nous proposons une méthode consistant à dessiner une sorte d'ombre planaire sous chaque tuile (cf. figure 6.11c). Chaque tuile est projetée telle une ombre sur

[1] http ://java.sun.com/products/jfc/tsc/articles/jcanyon/

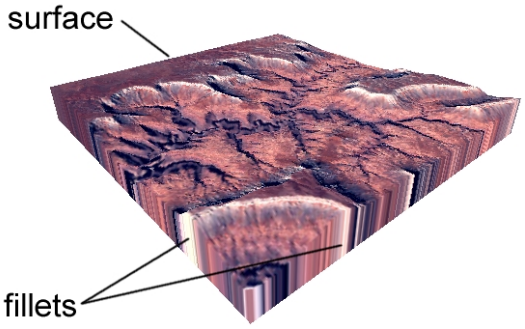


FIG. 6.9: *Technique du filetage.*

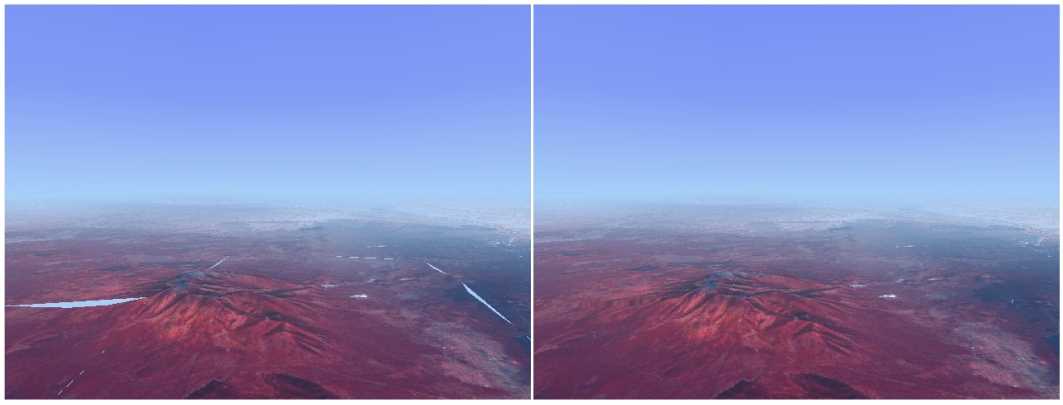


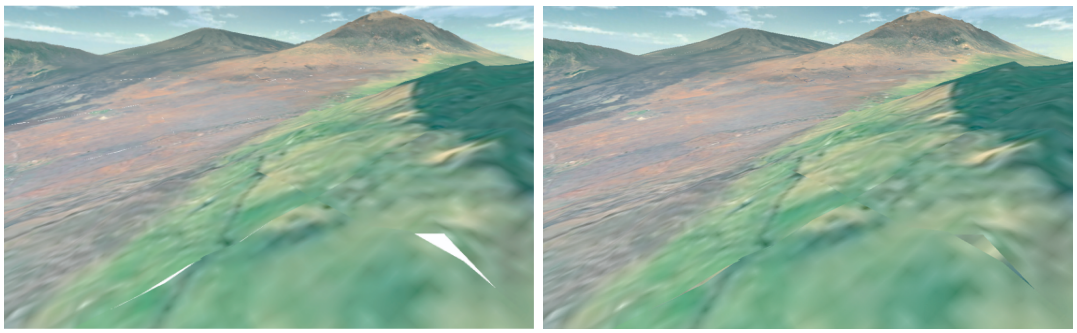
FIG. 6.10: *Surface obtenue sans et avec les filets.*

un plan situé sous la surface du terrain. L'ombre de la tuile est un trapèze pouvant être dessiné à l'aide de triangles. Cette ombre est texturée à l'aide de la même texture que la tuile correspondante. La position de l'ombre est calculée en projetant les coins de tuile à partir de la position du point de vue, de façon similaire au calcul classique des ombres planaires. Pour plus de détails sur les ombres planaires, le lecteur est invité à compulser [AMH02, pages 250–254]. Même si cette solution n'est pas parfaite et échoue dans certains cas, par exemple lorsque l'angle de vision est trop rasant sur la surface, elle est simple à implémenter et donne le plus souvent des résultats satisfaisants. La figure 6.11b montre le résultat obtenu en appliquant notre technique sur la figure 6.11a.

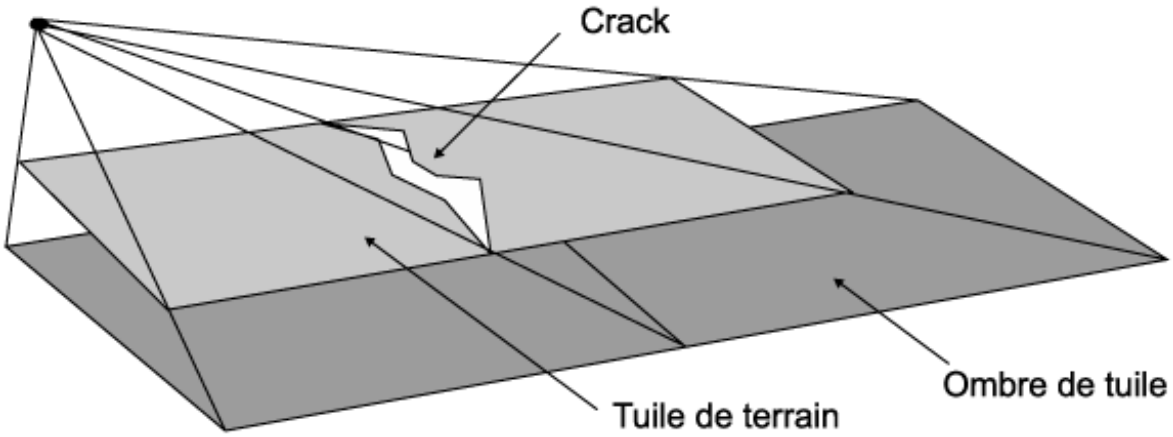
6.6 Extensions des nœuds VRML97

Nous gérons une base de données de fichiers au format VRML97 (texte ou binaire) décrivant une tuile de terrain. Notre implémentation est basée sur une amélioration du nœud VRML97 `ElevationGrid` que nous avons appelé `AutoElevationGrid`. Ce nœud a le même prototype que le nœud `ElevationGrid` mais son affichage se fait de façon adaptative selon l'algorithme précédemment décrit.

L'implémentation du pavage adaptatif est réalisée au sein d'un nouveau nœud appelé `AutoGrid25D`. L'accès à une tuile se fait par la détermination de son URL à partir d'un nom de base donné comme champ à ce nœud et des coordonnées de la tuile dans la grille.



a. b.



c.

FIG. 6.11: *Artefacts de cassure. a) Les cassures apparaissent sur le bords des tuiles lorsque les tuiles adjacentes ont des niveaux de détails différents. b) A l'aide d'un plan sous-jacent texturé projeté, l'effet de cassure est visuellement atténué. c) Deux plans d'ombres texturées sont rendus pour atténuer la discontinuité.*

Les textures sont gérées automatiquement en attachant une texture au champ `Material` de chaque tuile de géométrie.

6.7 Résultats

Dans cette section nous présentons des résultats expérimentaux obtenus avec notre technique. Afin de prouver les capacités d'adaptativité et de téléchargement progressif, nous avons réalisé des expériences sur des plates-formes très différentes : un PocketPC et une station de travail PC, avec différents modèles de terrains.

6.7.1 Grand Canyon sur PC

Dans un premier temps, nous proposons d'expérimenter notre méthode sur le modèle du Grand Canyon situé en Arizona, USA. Ce modèle, utilisé dans plusieurs travaux de recherche, a été introduit pour la première fois dans [Hop98]. Les données initiales [LGMA] sont constituées d'un MNA et d'une image satellite obtenus par l'USGS puis traités par Chad

McCabe du Microsoft Geography Product Unit. Le MNA est une grille de taille 4097×2049 (soit 8 394 753 points), la texture a une résolution de 4096×2048. L'espace entre chaque échantillon est de 60 mètres et la résolution des altitudes est de 10 mètres.

Ce modèle rentre complètement en mémoire sur notre plate-forme PC constituée d'un Pentium 4 (2.5GHz, 1Go de RAM, Quadro FX 500 128MB, AGP 8x). Hors-ligne, les données MNA et textures sont divisées en tuiles de taille 128×128. La base de données obtenue est encodée sous la forme de 561 fichiers VRML97 binaires zippés et autant de fichiers de textures au format JPEG. L'occupation disque de la base est de 26.2Mo. La taille moyenne d'une tuile du MNA est d'environ 50Ko et celle de la texture d'environ 15Ko.

Pour ce test, la machine cliente fait également office de serveur. La base de données se trouve donc localement sur le disque dur de la machine. Nous avons fixé le taux de rafraîchissement cible à 25tps, taux généralement considéré comme offrant une navigation suffisamment fluide et interactive. Différentes mesures obtenues lors d'un parcours sur le modèle sont présentées dans la figure 6.12. Le parcours peut être divisé en 4 tranches de temps :

0s – 13s. Le point de vue est initialement placé dans un coin du terrain et regarde l'intégralité du terrain. Nous attendons le chargement intégral du modèle avant d'effectuer le moindre déplacement. Le graphique du bas montre que les téléchargements sont distribués sur une période de 13s, en effet durant cette période le nombre de tuiles augmente quasi linéairement jusqu'à atteindre le seuil de 561. Comme le point de vue est dirigé de façon à voir l'intégralité du terrain, le nombre de tuiles rendues suit logiquement le nombre de tuiles chargées. Le graphique du haut montre que le taux de rafraîchissement converge rapidement vers le taux cible de 25tps. Le nombre de triangles affichés suit également cette règle. Les fluctuations du taux de rafraîchissement autour du taux cible sont dues à l'exécution en parallèle du processus léger s'occupant de décompresser les fichiers VRML97 et JPEG puis de les interpréter (incluant l'initialisation des tampons de vertex). Malgré ces fluctuations, on constate que le taux de rafraîchissement oscille autour du taux cible.

13s – 19s Nous attendons encore quelques secondes avant de démarrer la navigation. Nous pouvons voir que le taux de rafraîchissement est plus fidèle à la cible du fait que plus aucun téléchargement ne survient. On note également que le nombre de triangles rendus augmente rapidement d'environ 100000 triangles dès que le chargement cesse.

19s – 42s Nous entamons notre parcours et traversons le terrain dans sa longueur. Après 7 secondes de navigation lente, le taux de tuiles rendues décroît, du fait du *frustum culling* jusqu'à atteindre une valeur très basse lorsque nous atteignons le bord opposé du terrain. Le graphique du haut montre que le nombre de triangles affichés augmente massivement durant cette période et on constate clairement une inflexion à la 35ème seconde. Avant ce point d'inflexion, l'augmentation du nombre de triangles est dû au fait que moins de tuiles sont traitées, et donc moins de ressources CPU sont nécessaires lors de la traversée du graphe de scène (dont *frustum culling*, calcul des importances et allocation des budgets). Après la seconde 35, l'augmentation massive vient du fait que les listes d'affichages compilées restent en mémoire graphique, limitant ainsi les transferts mémoire RAM - mémoire graphique sur le bus graphique, En fait, l'inflexion survient lorsque moins de de 235 tuiles rendues. Or chaque tuile nécessite $128 \times 128 \times 4 \times 5 = 512$Ko de mémoire pour l'encodage du tampon de vertex (pour rappel, coordonnées 3D et textures) et $128 \times 128 \times 3 = 48$Ko pour la texture. En conséquence, la quantité de mémoire graphique globalement utilisée est de $(48 + 512) \times 235 = 128.51$Mo ce qui

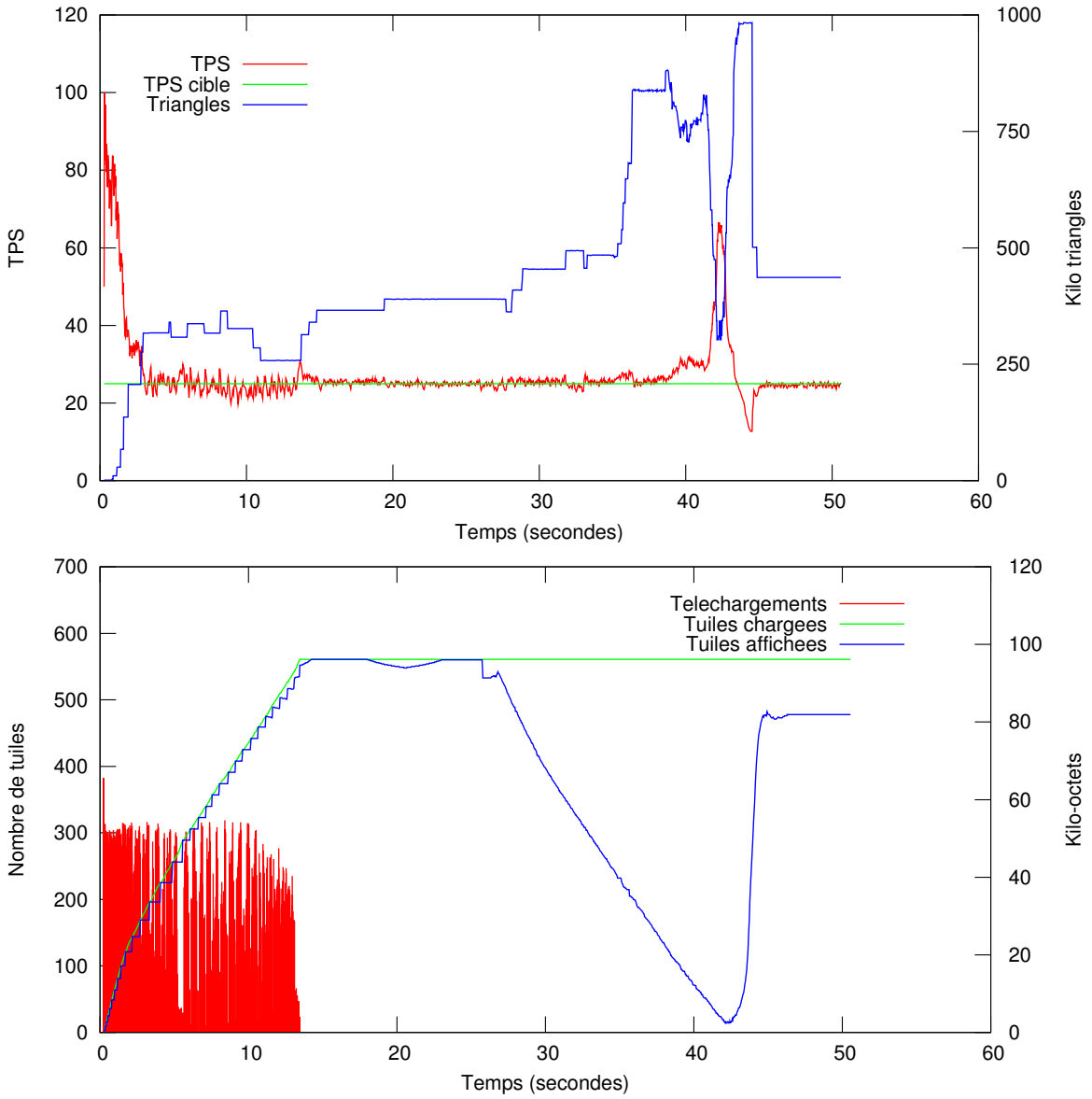


FIG. 6.12: *Performances mesurées durant une navigation sur le modèle du Grand Canyon. En haut : évolution temporelle du taux de rafraîchissement cible, le taux de rafraîchissement réellement atteint (TPS) et le nombre de triangles utilisés pour l'affichage de chaque trame. En bas : évolution temporelle des téléchargements (exprimés in Ko), le nombre de tuiles chargées et le nombre de tuiles affichées.*

correspond quasiment à la quantité de mémoire graphique (128Mo) dont dispose la carte 3D. Enfin, autour de la seconde 42, le taux de rafraîchissement augmente largement au dessus de 25tps du fait que les quelques tuiles affichées, même avec la plus résolution, contiennent un nombre de triangles bien inférieur au budget de triangles disponible.

42s – 50s Nous effectuons finalement un retournement de 180° suivi d'une montée en altitude

afin de visualiser la majeure partie du modèle. Les tracés du 1er graphique montrent une baisse du taux de rafraîchissement à 12tps durant une seconde environ. Cette durée correspond à la durée du filtre utilisée pour lisser son taux. Une fois la baisse du taux perçue, le système réagit immédiatement pour converger à nouveau vers le taux cible. Ce ralentissement massif est dû à l'estimateur du budget global d'Elkano qui avait estimé un budget très important lorsque le point de vue était sur le bord avant d'effectuer le retournement.

6.7.2 Puget Sound sur PC

Notre seconde expérimentation est effectuée sur le modèle d'un territoire situé dans la région du Mont Puget Sound dans l'état Washington State aux Etats-Unis. Les données utilisées ont été traitées par Lindstrom [LP01] à partir du modèle de l'USGS obtenu par l'Université de Washington. Le modèle est constitué de $16385 \times 16385 = 268468225$ échantillons espacés de 10m. Les altitudes sont échantillonnées sur 16bits à une résolution de 0,1m. Le MNA est disponible avec une texture artificielle calculée à partir des altitudes du terrain.

Cette fois-ci, le modèle est trop lourd (plus de 5Go de RAM seraient nécessaires) pour tenir entièrement dans la mémoire de la station PC utilisée (la même que précédemment). Comme pour le modèle du Grand Canyon, le MNA et la texture sont préalablement divisés en tuiles de taille (128×128). La base de données compressée obtenue consiste en 8192 fichiers (VRML97 et JPEG) et occupe un total de 60Mo sur le disque dur.

Pour ce test, nous utilisons les mêmes paramètres et la même configuration que pour l'expérience précédente. La figure 6.13 présente les mesures enregistrées durant un parcours sur l'ensemble du modèle, allant d'une frontière à une autre et en prenant parfois de l'altitude pour obtenir une vue d'ensemble. Le graphique du haut nous permet d'observer des variations similaires que celles enregistrées sur le Grand Canyon. Dans le graphique du bas, nous avons ajouté la quantité de mémoire utilisée durant le trajet. On constate clairement que le tracé de cette variable est directement corrélé avec le nombre de tuiles. Les résultats montrent également que l'utilisateur conserve toujours une bonne interactivité même lors de la réception de nouvelles données.

6.7.3 Puget Sound sur PocketPC

Afin de valider notre solution de visualisation, nous avons effectué la visualisation de ce même modèle du Puget Sound sur un PocketPC de type Toshiba e800. Ce PDA est cadencé à 400MHz, dispose de 64Mo de mémoire RAM mais pas de processeur graphique dédié. La bibliothèque OpenGL|ES utilisée est donc une implémentation entièrement logicielle (et donc repose totalement sur le CPU). La résolution de l'écran est de 320×240 points (voir figure 6.1). En guise de connexion avec le serveur PC (en l'occurrence la même machine que cité précédemment), nous utilisons une connexion filaire USB2.0. Le parcours effectué sur le modèle consiste à aller d'un angle à son opposé, en suivant une direction diagonale.

La figure 6.14 montre les résultats mesurés avec un taux de rafraîchissement cible fixé à 7tps (cette valeur d'apparence faible est, sur ce matériel, suffisante pour offrir à l'utilisateur une sensation d'interactivité) et nous requérons que 20Mo de mémoire soient toujours disponibles. Comme nous pouvons le constater sur le graphique du haut, le système s'adapte assez bien pour atteindre le taux de rafraîchissement désiré. Cependant, on constate des fluctuations plus importantes lors des phases de téléchargement effectuées en parallèle du rendu

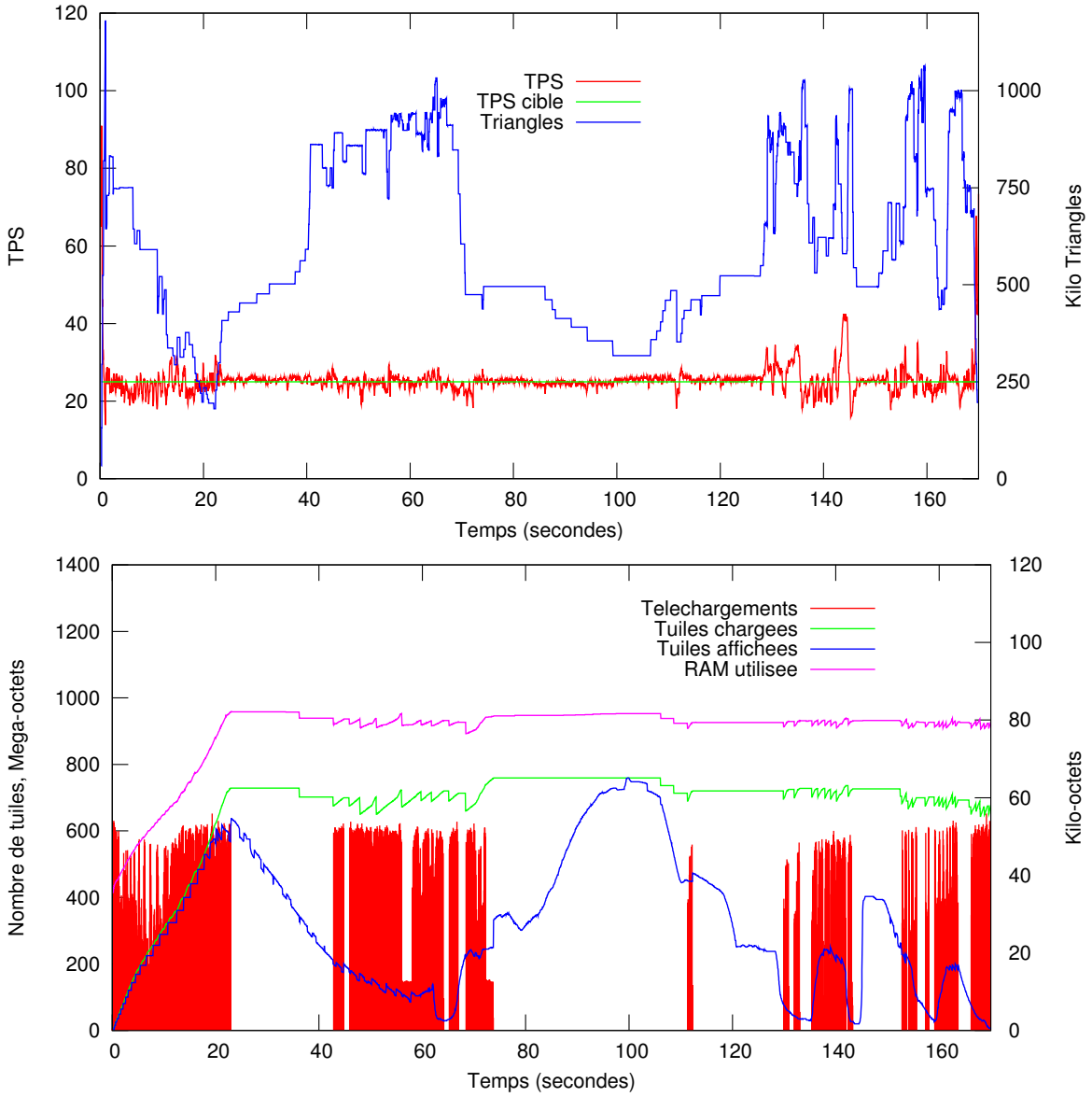


FIG. 6.13: *Performances mesurées durant une navigation sur le modèle du Grand Canyon sur PC. En haut : évolution temporelle du taux de rafraîchissement cible, le taux de rafraîchissement réellement atteint (TPS) et le nombre de triangles utilisés pour l'affichage de chaque trame. En bas : évolution temporelle des téléchargements (exprimés en kilo-octets), le nombre de tuiles chargées et le nombre de tuiles affichées. Nous avons également ajouté ici la quantité de mémoire utilisée, quantité corrélée avec le nombre de tuiles mémorisées.*

(voir le graphique du bas). Ces fluctuations sont dues à ce que, cette fois, l'ensemble des traitements est réalisé sur le CPU (de la décompression à la rastérisation des triangles). On constate également qu'on atteint une limite maximale de 10000 triangles par image, lorsqu'aucun téléchargement n'est effectué. Cependant, cette limite est relativement bonne si nous rappelons une fois de plus que la machine de faible capacité ne dispose que d'un CPU

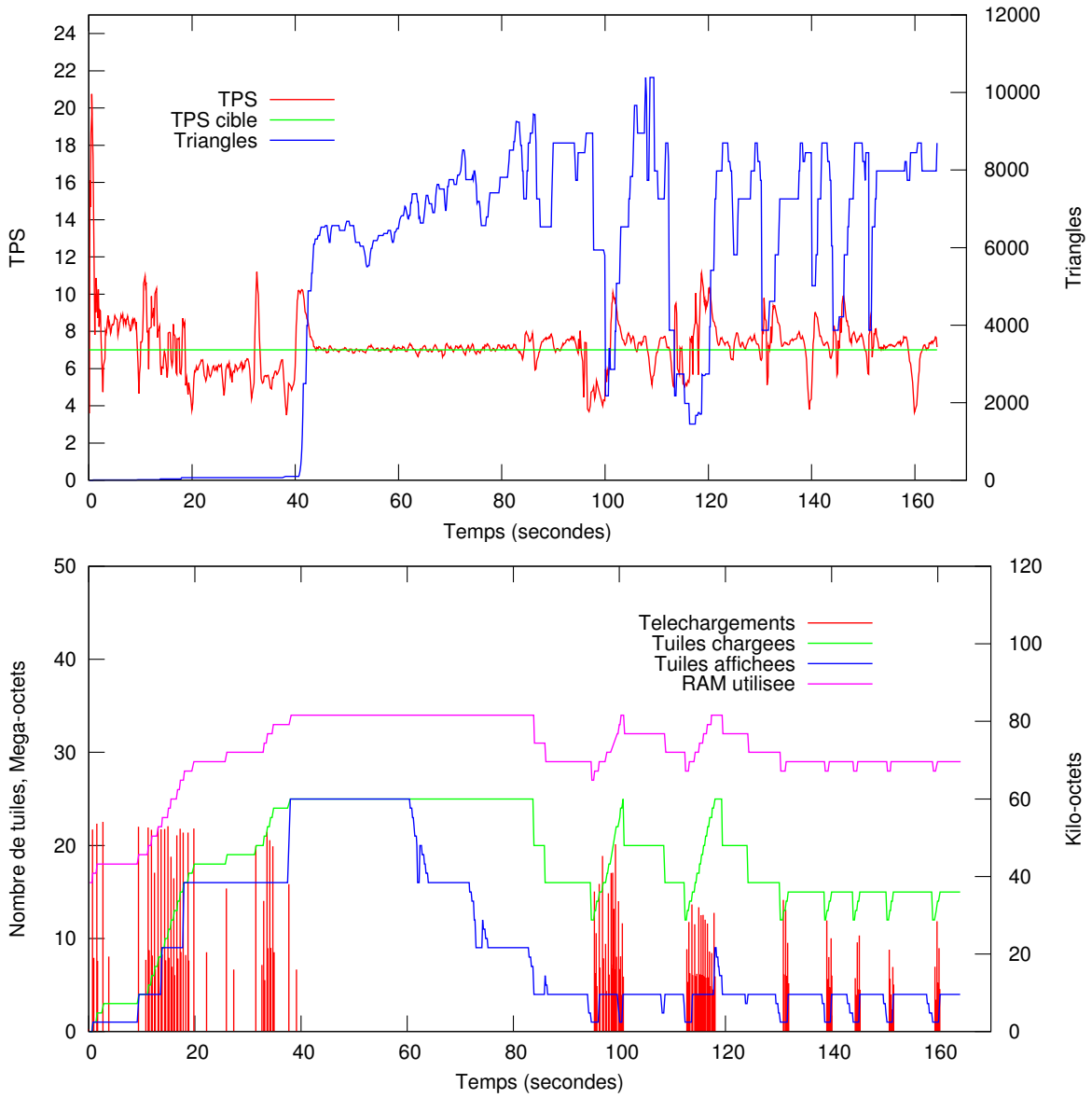


FIG. 6.14: *Performances mesurées durant une navigation sur le modèle du Grand Canyon sur PocketPC. Les valeurs mesurées sont les mêmes que sur la figure 6.13.*

et que l'arithmétique y est effectuée uniquement en valeurs entières. Il faut en effet noter que dans notre implémentation, tous les calculs en virgule flottante sont émulés de façon logicielle et que notre code n'a pas été spécialement optimisé pour transposer certains calculs en virgule fixe.

6.8 Extensions

FIG. 6.15: *Téléchargement progressif et rendu adaptatif de grands terrains. Modèle du Bugaboos (8 millions de triangles et 208MB of textures) visualisé sur un PocketPC VGA 640x480 avec un taux de rafraîchissement fixé à 2tps et connecté à l'aide d'une connexion WiFi.*

6.8.1 Téléchargement progressif des tuiles

Afin d'accélérer le chargement des tuiles et d'augmenter la quantité de tuiles chargées en mémoire, nous avons décidé d'effectuer un téléchargement progressif des niveaux de tuiles plutôt que de recevoir d'un seul coup l'intégralité du MNA la décrivant. Cette approche utilise la capacité d'Elkano consistant à permettre le développement de nœuds progressifs à l'aide d'une communication entre le nœud existant côté client et son homologue crée côté serveur. L'idée consiste alors à ce que le client télécharge au besoin les vertex nécessaires à l'affichage d'un niveau de *strip-mask* plus détaillé. Afin d'augmenter la vitesse de réponse du serveur, nous avons mis au point un format de fichier spécifique où les points nécessaires pour passer du niveau n au niveau $n+1$ sont stockés de façon contiguë. Une seule opération de lecture est alors nécessaire pour lire les points à envoyer au client. Lorsque le client reçoit les points, il les ajoute dans un tableau de vertex et augmente le compteur décrivant le niveau de masque maximum disponible pour cette tuile.

Les premiers résultats obtenus (voir figure 6.15) avec ce nouveau niveau de *streaming* localisé sont prometteurs. Cependant, il reste à déterminer une politique de libération des niveaux : par exemple lorsqu'au bout d'un certain nombre de trames, le niveau de masque maximum m n'a pas été utilisé, celui-ci est décrémenté et la mémoire utilisée pour stocker les points est libérée. Il faudra également étudier à partir de quel moment, fonction des ressources disponibles, de la bande passante et de la latence du réseau, il peut être utile d'utiliser ce niveau de *streaming*.

6.8.2 Géovisualisation

Avec la disponibilité de bases de données d'informations géographiques mondiales, des outils de visualisation grand public de la surface terrestre extrêmement efficaces sont apparus ces dernières années (en particulier *NASA WorldWind*, *KeyHole* devenu *Google Earth* ou encore le *GéoPortail* de l'IGN). Toutes ces applications proposent des solutions de visualisation de MNA de la terre (le plus souvent les données de la mission SRTM) et de navigation totalement géoréférencées. L'utilisateur peut alors accéder directement à un point du globe à l'aide de ses coordonnées géographiques.

Nous avons décidé d'associer ce mécanisme à notre solution de rendu de terrains. Notre plate-forme de visualisation Elkano étant compatible avec VRML97, il nous est paru naturel de lui adjoindre un support pour l'extension GeoVRML.

Le GeoVRML [RIL00] est une spécification proposée par le Consortium Web3D en 1998 définissant des extensions à VRML97 pour permettre la création d'applications géographiques. Concrètement, le GeoVRML est un ensemble de 10 nouveaux nœuds permettant de géoréférencer les objets dans l'espace. En particulier, le nœud `GeoElevationGrid`, qui étend le nœud VRML `ElevationGrid`, permet de générer, à une position géographique donnée (latitude, longitude), le maillage d'un terrain à partir d'un MNA. La géométrie d'un tel terrain prend alors automatiquement forme sur l'ellipsoïde terrestre. GeoVRML a été proposé et accepté comme partie intégrante du récent langage X3D (eXtensible 3D, format de description de scènes 3D créé par le consortium Web3D dans le but de succéder à VRML97 et normalisé par l'ISO en 2005).

L'implémentation du GeoVRML au sein d'Elkano a été également l'occasion pour nous de permettre la lecture des fichiers au format X3D. Grâce à ces extensions, nous avons pu très facilement créer une application de géovisualisation du type *TerraVision* exploitant notre technique de rendu adaptatif.

6.9 Bilan

Dans ce chapitre, nous avons présenté une solution permettant le téléchargement progressif et la visualisation temps-réel de grands MNA texturés. Tandis que la plupart des approches classiques se concentrent sur l'optimisation en temps réel du maillage de façon très locale, notre approche tend à alléger les calculs sur le CPU et la consommation mémoire en transférant la charge sur le processeur graphique 3D (ou son émulation logicielle). Autour d'un algorithme de pavage et une structure de données multi-résolution par tuile, nous avons proposé une technique adaptative en regard des capacités de la machine client. D'un côté, la gestion dynamique des tuiles basée sur une adaptation mémoire permet un téléchargement progressif des données (géométrie et images de texture). Ce mécanisme permet à l'utilisateur de naviguer immédiatement dans l'environnement virtuel. D'un autre côté, les tuiles sont rendues efficacement en utilisant un ensemble de masques précalculés représentant les indices d'une chaîne de triangles. La résolution des tuiles est choisie selon des paramètres globaux et locaux ainsi qu'en fonction des capacités de la carte graphique 3D dans le but d'atteindre un taux de rafraîchissement interactif donné. Les résultats que nous avons présentés attestent de la robustesse de l'adaptation obtenue.

Dans les extensions futures possibles, nous envisageons d'utiliser une structure de données multi-résolution permettant le téléchargement progressif de chaque tuile. De cette façon, les niveaux de tuiles pourraient être téléchargés uniquement en cas de besoin. Cette répartition

du téléchargement dans le temps permettrait de télécharger plus rapidement les tuiles visibles proches à de bonnes résolutions, tandis que les tuiles lointaines seraient téléchargées uniquement à de faibles résolutions. Une autre amélioration serait d'éviter d'une façon plus efficace encore les trous dus à la discontinuité de la surface. Nous pensons que ce problème pourrait être résolu à l'aide d'une autre définition des masques, prenant en compte des zones de transition pour assurer une continuité entre les tuiles adjacentes.

Chapitre 7

Visualisation de terrains à distance

7.1 Introduction

Dans ce chapitre nous présentons deux techniques de visualisation de scènes 3D géographiques que nous avons mises au point et qui sont basées sur un modèle client / serveur dans lequel le serveur synthétise l'image tandis que le client léger est utilisé comme moyen d'affichage et d'interaction uniquement. Ces techniques permettent de répondre efficacement aux problèmes de ressources mémoire et calculatoires des assistants personnels ou téléphones portables disposant d'une connexion à un réseau sans fil.
Dans la section 7.2 nous présentons une technique totalement connectée dans laquelle le serveur génère au fur et à mesure le flux d'images synthétisées en fonction des manipulations effectuées par l'utilisateur qui lui sont transmises.
La section 7.3 présente l'ébauche d'une technique de repérage offrant à l'utilisateur en situation de mobilité de type randonnée, un aperçu du panorama l'environnant ainsi que des données contextuelles pouvant l'intéresser. Dans ce cas le protocole utilisé est non connecté : le client fait une requête en fournissant au serveur ses coordonnées et celui-ci lui retourne un ensemble de données (images et méta-données) correspondant à sa position.

7.2 Rendu à distance

7.2.1 Introduction

Le rendu/visualisation 3D interactive à distance (en anglais *remote rendering*) dans un cadre de mobilité n'est pas une technique nouvelle. Différents travaux l'ont exploité dans différents contextes.

Au début des années 1990, la bibliothèque Distributed IRIS GL de SGI (*Integrated Raster Imaging System Graphics Library* ancêtre de la bibliothèque OpenGL) est l'équivalent de l'actuel protocole GLX d'OpenGL : les commandes graphiques GL sont interceptées par la bibliothèque cliente, envoyées par le réseau au serveur et transmises au hardware du serveur pour y être exécutées. Le produit OpenGL Vizserver de SGI permet de créer une session de rendu à distance entre un ordinateur de bureau et des serveurs graphiques de type SGI Onyx.

Engle *et al.* [ESEE99, ESE00] ont développé une interface pour les applications OpenInventor et Cosmo3D. L'idée consiste à envoyer via un réseau les images calculées et compressées

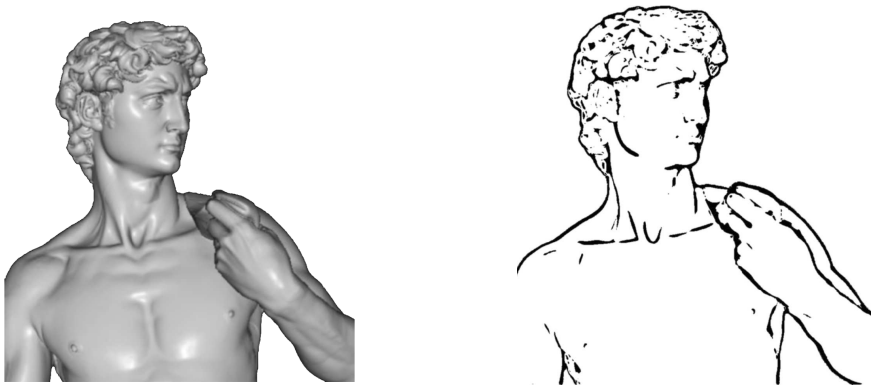


FIG. 7.1: *Modèle du* David *de Michel-Ange rendu de façon dite "réaliste" (à gauche) et en utilisant un rendu NPR à base de traits caractéristiques (à droite).*

sur le serveur à un client disposant d'une machine virtuelle Java. Les événements générés par le client étant retournés comme des requêtes CORBA.

Dans [DGE04], Diepstraten *et al.* proposent une technique reposant sur la simplicité et la facilité de compression des images obtenues à l'aide d'un rendu dit non-photoréaliste (NPR). Contrairement au rendu d'image classique (abusivement appelé photoréaliste pour s'opposer au NPR) qui cherche à simuler au mieux les phénomènes lumineux (lumière, ombres, caustiques, etc.) et les propriétés des matériaux des objets du monde réel, les techniques NPR cherchent à reproduire les techniques de reproduction expressives utilisées dans les arts plastiques. Il existe de très nombreuses techniques de rendus NPR, pour plus de détails nous renvoyons le lecteur à l'ouvrage de Strothotte et Schlechtweg [SS02]. Une famille de méthode cherche à créer des images se rapprochant des crayonnés de la bande dessinée en générant uniquement des lignes caractéristiques du modèle affiché (voir figure 7.1. Diepstraten *et al.* génèrent sur, le serveur, une image de ce type en noir et blanc. L'image obtenue est vectorisée en un ensemble de traits. Ces traits sont ensuite compressés et transmis au client qui n'a plus qu'à les rasteriser sur sa surface d'affichage.

La technique que nous avons mise au point est assez similaire à cette dernière. Elle diffère cependant en deux points : nous transmettons l'image binaire compressée plutôt qu'un ensemble de segments et le NPR n'est utilisé que lors de la phase d'interaction.

7.2.2 Schéma proposé

Notre technique repose sur une application répartie communiquant sur un réseau haut débit à l'aide du protocole TCP/IP. L'application se divise en deux parties :

1. d'une part la partie client, généralement exécutée sur un terminal mobile à faibles capacités de calcul (PDA, téléphone portable, etc.) qui permet à l'utilisateur de visualiser et d'interagir avec la scène disponible sur le serveur qu'il a préalablement choisie ;
2. d'autre part la partie serveur située sur une machine disposant d'une bonne capacité de traitement, en particulier graphique.

Le serveur stocke ou a accès à une base de données de scènes 3D (dans notre cas en OpenInventor, VRML97 ou X3D). Lorsque la connexion avec le client est établie, le serveur transmet la liste des scènes disponibles au client. Une fois la scène choisie par l'utilisateur, le

client envoie au serveur le nom de la scène 3D à visualiser. Dès la scène ouverte sur le serveur, celui-ci rentre dans la phase de rendu-interaction illustré par le diagramme 7.2. Le principe de notre technique consiste à ce que le serveur puisse alterner entre deux schémas de rendus en fonction de l'état du client. Nous distinguons ainsi les phases d'interaction et les phases statiques.

7.2.2.1 Phases d'interaction

Les phases d'interaction sont les moments durant lesquels le client est en train d'interagir avec la scène sur les dispositifs d'interaction du terminal mobile (les touches et l'appui stylet sur l'écran tactile du PDA dans notre cas). Lors de ces phases, il est important que l'utilisateur puisse constater en temps réel l'effet de ses actions sur la scène. Il existe donc une contrainte de performance forte sur la requête du client fournissant au serveur les informations sur l'interaction réalisée et attendant en retour la nouvelle image à afficher. Il convient donc de réduire sensiblement la quantité de données transmises sur le réseau et, si nécessaire, le temps de calcul de la scène par le serveur (par exemple avec les techniques décrites dans le chapitre précédent). Une image couleur de taille 320×240 (résolution dite QVGA utilisée par de nombreux PDA) codée en 16 bits sous forme brute non compressée afin d'économiser le temps de compression / décompression pèse 150Ko. Dans la pratique, le temps de calcul et de transmission d'un tel flux d'images en temps réel s'avère difficile à atteindre, car elle nécessite un haut débit (pour 15 images par secondes, 2,2Mo de données doivent transiter par seconde) et une latence (le temps minimum pour effectuer un aller et retour entre deux points du réseau, la latence est forcément longue si les deux points sont éloignés) faible, ce qui est rarement le cas sur des réseaux sans fil.

Nous proposons donc de transmettre une image en noir et blanc décrivant les traits ca-

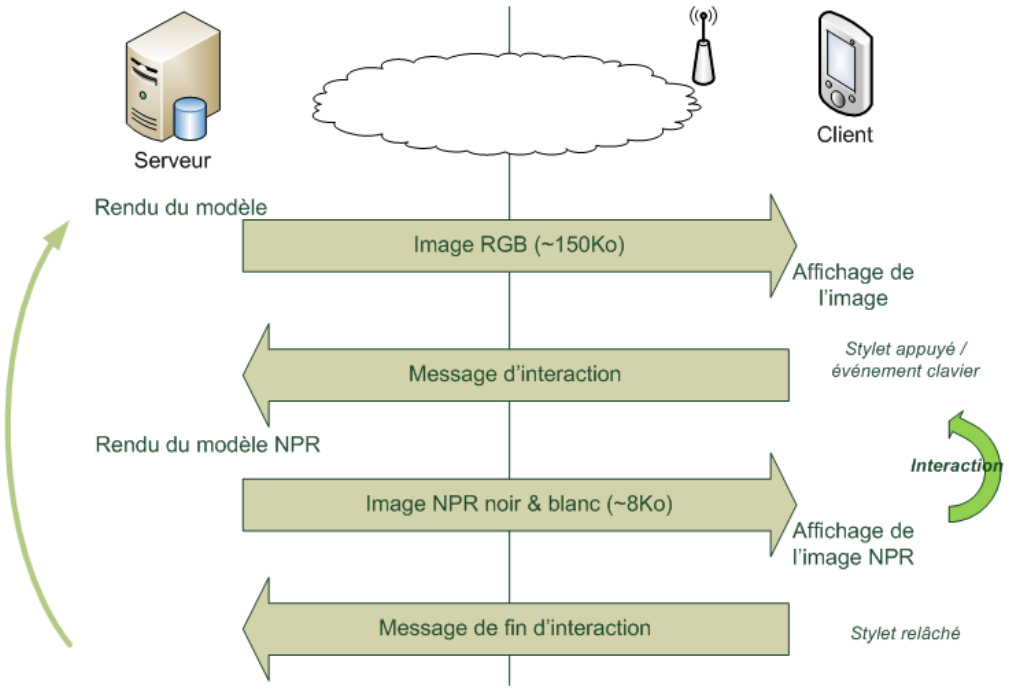


FIG. 7.2: *Diagramme décrivant notre protocole de rendu à distance.*

ractéristiques de la scène visualisée. Pour obtenir un tel rendu, nous utilisons une technique de rendu NPR. Dans notre cadre expérimental, nous utilisons une technique consistant à extraire de la scène différents contours caractéristiques tels que les frontières et les silhouettes des objets.

Contrairement à [DGE04], nous ne vectorisons pas l'image pour en extraire un ensemble de lignes. En effet, si cette vectorisation a un coût, il faut aussi compter que la rastérisation des lignes en a un également côté client. Or il est généralement moins coûteux de transférer d'un seul coup à la mémoire graphique un tampon image plutôt que de modifier la valeur des pixels un à un. Aussi, l'image binaire calculée est plutôt encodée à l'aide d'une technique de type *codage des répétitions* (en anglais *run-length encoding*, ou RLE). En moyenne, pour une image au format QVGA, le fichier obtenu est de l'ordre de moins d'une dizaine de kilos octets.

7.2.2.2 Phases statiques

Les phases statiques correspondent aux périodes durant lesquelles l'utilisateur n'interagit plus avec la scène. Si la scène n'est pas animée, il n'y a donc plus de contrainte prégnante concernant le temps de calcul et de transfert de l'image entre le serveur et le client. Pour cette phase, la scène peut être calculée en couleur avec une technique de rendu classique.

7.2.3 Résultats et bilan

Nous avons expérimenté cette technique sur un PDA Toshiba e800 disposant de connexions USB 2.0 et WiFi 802.11b (débit théorique de 11 Mbps, 6 Mbps réels, avec une portée pouvant aller jusqu'à 300 mètres dans un environnement dégagé). Sur notre jeu de scènes 3D, nous avons obtenu une bonne interactivité, offrant un taux de rafraîchissement de l'ordre de 15tps sur le PDA, taux largement suffisant sur une telle machine. La quantité de données transitant sur le réseau pendant les phases d'interaction est relativement faible : pour 15 trames par secondes, 15×10=150Ko par seconde. La figure 7.3 illustre les deux phases de notre technique.

En conclusion, cette approche de rendu à distance en deux phases est intéressante pour visualiser, sur des machines à faibles ressources, des données complexes avec des techniques de rendu éprouvées sur des machines puissantes. La visualisation en NPR lors des phases d'interaction permet de baisser sensiblement la quantité de données transmises sur le réseau et d'augmenter par là même la sensation d'interactivité de l'utilisateur. En contrepartie, cette technique nécessite une très bonne accessibilité au réseau et en particulier une latence faible. Elle est donc adaptée à un environnement spécifique tel que l'enceinte d'un musée ou d'un site touristique, et, pour pouvoir être utilisée à simultanément par de nombreux utilisateurs, nécessite une puissante grappe de serveurs de rendu dédiés.

7.3 Service de panoramas instantanés

Les solutions précédemment présentées permettent une navigation virtuelle 3D en temps réel dans une scène géographique sur de petits terminaux connectés. Nous l'avons dit, cette possibilité est importante et peut être utile dans de nombreuses applications. Dans cette section, nous présentons une approche différente, s'insérant dans le cadre d'une application de repérage où le but est de fournir à l'utilisateur des informations sur ce qu'il voit autour de lui plutôt que ce qu'il pourrait voir s'il avançait dans telle ou telle direction.

FIG. 7.3: *Rendu à distance. A gauche : rendu non-photoréaliste (léger à transmettre) lors des phases d'interaction. A droite : rendu classique en couleur lorsqu'il n'y a plus de mouvement.*

Avec l'essor des technologies de positionnement telles que le GPS ou bientôt l'alternative européenne Galiléo, de nombreuses applications cartographiques embarquées ont vu le jour afin de fournir à l'utilisateur un service de repérage sur des cartes 2D. Le plus généralement, un capteur GPS fournit la position en temps réel de l'utilisateur à l'application qui affiche alors cette position sur une carte (routière le plus souvent). Les données cartographiques utilisées sont généralement stockées localement dans une carte d'extension mémoire. Dans de nombreux cas, cette technique de repérage est très efficace, notamment dans le cas de parcours automobiles. Cependant, une vue en 2D, même si elle est présentée en vue oblique, n'est pas forcément efficace dans le cas d'un parcours pédestre de type randonnée en haute montagne.

FIG. 7.4: *Exemple de visualisation obtenue avec le service de panoramas instantanés.*

Dans cette section, nous présentons l'ébauche d'une technique que nous avons imaginée

offrant à l'utilisateur mobile un repérage instantané par le biais d'un service de panoramas virtuels en 3D. A noter que cette technique à fait l'objet d'une présentation / démonstration dans un séminaire associé à CHI 2007 [HPKG07].

L'application mise au point est particulièrement adaptée à un randonneur ou à un touriste évoluant dans un site accidenté. Notre approche client / serveur en mode non connecté – c'est-à-dire nécessitant une connexion ponctuelle, le temps de la requête client et de la réponse serveur – repose sur les services de bases de données géographiques et le rendu de terrains en 3D, notamment à l'aide de techniques non-photoréalistes. Le résultat du traitement par le serveur est une scène interactive légère au format X3D pouvant être visualisée avec n'importe quel navigateur X3D – et en particulier avec la plate-forme Elkano –, voir figure 7.4.

7.3.1 Applications GPS

De nombreuses applications commerciales de géopositionnement permettent déjà à l'utilisateur disposant d'un terminal équipé d'un récepteur GPS / DGPS de visualiser leur position sur des cartes topographiques ou routières et de lui fournir d'autres informations contextuelles (le plus souvent commerciales) sur son environnement. Généralement, les données géographiques spécifiques sont vendues avec le logiciel.

La disponibilité grandissante des MNA et de bases d'informations géoréférencées acquis ou générés par les méthodes décrites dans la première partie de cette thèse a mené au développement de logiciels pour station de travail dédiés au randonneur. Le but de ces outils est généralement de visualiser des informations cartographiques (calcul de dénivelés, distances, etc.) dans le but de préparer une excursion. Une autre application est de fournir une information touristique des objets alentours à un terminal mobile équipé d'un récepteur GPS à travers la scène 3D préalablement modélisée d'un site donné [BC05] .

7.3.2 Panoramas virtuels interactifs

Le but de cette technique n'est pas de fournir à l'utilisateur un environnement 3D totalement interactif dans lequel il pourrait naviguer comme dans les techniques présentées précédemment. Il s'agit plutôt de restreindre la scène 3D à une vue panoramique du paysage alentour augmentée d'informations contextuelles potentiellement utiles à l'utilisateur. Le pipeline général de notre approche est illustré par la figure 7.5 et peut être résumé par les étapes suivantes :

- L'application cliente détermine la position actuelle (latitude et longitude) de l'utilisateur à l'aide du capteur GPS et l'envoie au serveur.
- A partir de cette position le serveur principal acquiert, si besoin, les informations pertinentes dans différentes bases de données (MNA, toponymes, et données vectorielles diverses) distribuées sur différents services Web OGC comme Web Map Service et Web Feature Service.
- Le serveur principal demande au serveur graphique de générer un panorama virtuel centré à la position de l'utilisateur en calculant une scène 3D avec les données topographiques du MNA.
- Le serveur principal génère une scène X3D utilisant les images du panorama calculées et d'autres nœuds correspondants aux méta informations.
- Finalement, la scène X3D compressée est envoyée au client mobile. Une fois la communication terminée, l'utilisateur peut visualiser la scène panoramique.

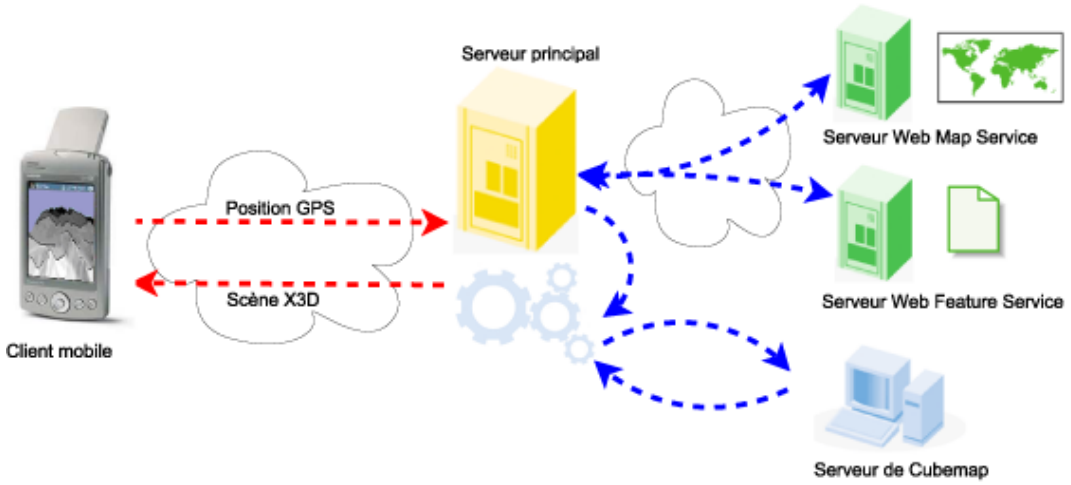


FIG. 7.5: *Diagramme illustrant le service de panorama.*

Cette approche possède de nombreux avantages :

- contrairement aux approches de rendu client / serveur, une connexion permanente à haut débit n'est pas nécessaire. En effet, dans cette configuration la connexion est limitée à un échange requête / réponse et peut donc transiter à faible coût via les techniques GSM ou GPRS actuelles ;
- des machines disposant de faibles ressources de calcul (tels que des téléphones portables) peuvent en tirer profit, rendant cette technique facilement exploitable commercialement ;
- les données 3D originales ne sont pas envoyées au client. Cette limitation peut être intéressante pour les sociétés souhaitant conserver le contrôle sur leurs données ;
- le rendu 3D du modèle de terrain peut être fait en utilisant les capacités de serveurs dédiés disposant de cartes graphiques puissantes et programmables à l'aide de techniques de rendu éprouvées.

Nous décrivons maintenant les différents choix techniques effectués pour mettre en œuvre notre approche.

7.3.3 Construction du panorama

A partir d'une position donnée et d'une étendue donnée, le serveur télécharge le MNA environnant en accédant à des serveurs OGC Web Map Service ainsi que des données vectorielles additionnelles telles que les cours d'eau, les zones forestières ou les chemins de randonnée à l'aide de serveurs OGC Web Feature Service. Les données obtenues sont transmises à un serveur de rendu local pour effectuer la génération du panorama. Différentes techniques de rendu peuvent être alors utilisées. En particulier, nous pensons qu'un rendu expressif de type NPR permettant de faire ressortir les lignes caractéristiques de la surface topographique (contours, lignes de crêtes, etc.) est particulièrement adapté pour améliorer la lisibilité du panorama Une analogie peut être faite avec les cartes topographiques 2D où l'information symbolique est plus utile à la compréhension qu'une image satellite pour une tâche de localisation. Ceci est particulièrement vrai dans le cadre d'une configuration de visualisation en mobilité (petits écrans, conditions lumineuses, etc.). La figure 7.6 illustre des exemples de rendus de terrain

NPR.

FIG. 7.6: *A gauche : terrain rendu avec une texture topographique. A droite : terrain rendu en utilisant un programme de shader NPR.*

La création du panorama s'effectue en capturant les 6 images visualisées à travers les faces d'un cube virtuel centré à la position de l'utilisateur comme illustré dans la figure 7.7. Les 6 images obtenues sont ensuite utilisées pour paramétrer un nœud d'arrière-plan *Background* dans la scène X3D.

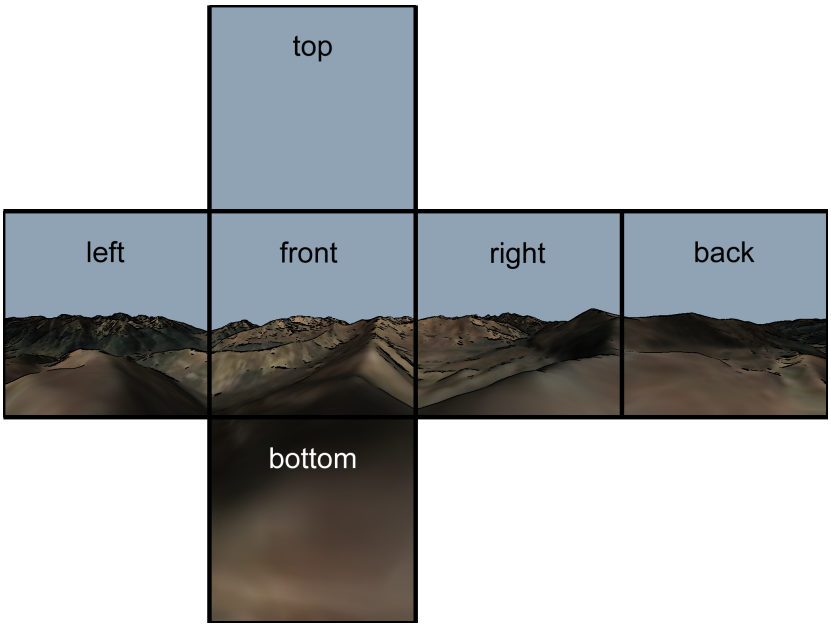


FIG. 7.7: *Les 6 faces d'un cube virtuel constituant l'arrière-plan de la scène panoramique.*

7.3.4 Ajout d'informations additionnelles

En plus du panorama, le serveur acquiert des données décrivant des informations contextuelles. Il peut s'agir par exemple, du nom et de la hauteur des montagnes environnantes ou de la position des refuges alentour pouvant intéresser le randonneur. A cause de la petite taille de l'écran du terminal mobile, nous proposons de ne pas visualiser ces informations sur les images de panoramas générées. De façon alternative, nous générons des nœuds d'information à l'aide du langage X3D. Visuellement, un tel nœud est représenté sous la forme d'une petite sphère semi-transparente positionnée dans la scène 3D. Si l'utilisateur la sélectionne, une zone de texte s'affiche avec les informations relatives à l'objet décrit. Dans la pratique, nous avons introduit un nouveau nœud appelé `Info` à la d'un prototype externe (`EXTERN PROTO` en X3D).

7.3.5 Interaction côté terminal

Une fois la scène panoramique interactive construite par le serveur, celle-ci est envoyée au client. Il reçoit alors la scène X3D compressée comprenant les 6 textures constituant le panorama ainsi que les nœuds contextuels supplémentaires. N'importe quel navigateur X3D (par exemple MobiX3D ou Pocket Cortona) permet alors d'effectuer la visualisation des données. Afin de faciliter l'orientation, une boussole virtuelle peut être ajoutée à la scène 3D et une option permet d'afficher la carte topographique 2D afin d'augmenter la perception d'orientation.

En utilisant le stylet ou une autre technique de sélection, l'utilisateur peut bénéficier de la vue augmentée à 360° de son environnement. En cliquant sur une sphère interactive, il peut afficher ou masquer les informations contextuelles additionnelles potentiellement utiles à son parcours.

7.3.6 Conclusion et travaux futurs

Nous avons présenté un schéma général pour le développement d'un service de repérage en 3D pour terminaux mobiles basé sur des services Web. Basé sur des solutions techniques simples, efficaces et standardisées, un panorama virtuel interactif représentant le paysage alentour de l'utilisateur est proposé à la demande à l'aide d'une simple requête à un serveur distant et d'un navigateur 3D.

Dans des travaux à venir, nous réaliserons un prototype plus abouti. Il pourrait être intéressant pour l'utilisateur de coupler en temps réel son orientation avec celle du panorama virtuel visualisé. Enfin, nous comptons identifier les styles de rendus expressifs les plus adaptés pour ce genre de tâches et les valider par des études utilisateurs.

Troisième partie

Interaction sur terminaux mobiles

Les récentes avancées techniques des ordinateurs de poche permettent d'offrir à l'utilisateur des applications interactives traitant des données complexes 2D mais aussi 3D sur de simples assistants personnels ou des téléphones portables. Ces capacités permettent d'imaginer de nouvelles applications pour la vie quotidienne, qu'elles soit professionnelles ou personnelles. Par exemple, un technicien sera en mesure de visualiser et de manipuler sur site le modèle en 3 dimensions du moteur qu'il est en train d'inspecter. Un archéologue pourra comparer sur place l'état d'un vestige avec le monument préalablement reconstruit par ordinateur et ce, à l'aide de son TMC, etc. D'une manière générale, l'étude des différentes façons d'interagir avec un ordinateur appartient au domaine de l'IHM (interfaces homme-machine). Dans le cadre de nos recherches sur la visualisation de modèles de terrains sur ce type de machines, nous nous sommes intéressés aux interfaces permettant à l'utilisateur de naviguer dans ces scènes.

Différents challenges sont à relever avec le développement des ordinateurs de poche en termes de capacités de calculs, de techniques de visualisation temps-réel, mais aussi en terme d'interfaces utilisateurs. Dans cette partie nous nous intéressons particulièrement à cette dernière problématique.

L'interaction sur TMC

Même si les techniques de pointage direct par stylet semblent bien acceptées pour l'interaction avec un TMC, nous verrons qu'elles ne sont pas toujours la meilleure solution, en particulier quand la visualisation est importante.

Contraintes

Comparés au ordinateur de bureau, les ordinateurs de poche ont certaines caractéristiques qui doivent être prises en compte pour le développement d'interfaces utilisateurs adaptées.

Visualisation limitée

La première caractéristique est la taille réduite de l'écran, principale limitation sur ce type de machine. En conséquence, la zone de visualisation ne doit pas être trop occultée par la main de l'utilisateur lors de l'utilisation du stylet ou l'appui sur une touche afin d'assurer une vision complète. De plus, l'affichage de l'écran généralement de type TFT ne peut être perçu que dans un intervalle d'angles de vue, c'est-à-dire quand l'utilisateur fait face à l'ordinateur. Aussi l'interaction doit pouvoir se faire sans avoir à modifier sensiblement l'orientation de l'écran.

Utilisation mobile

Les TMC sont destinés à une utilisation en situation de mobilité. En conséquence, les interfaces d'interaction doivent pouvoir s'opérer dans ces conditions. En particulier, elles doivent pouvoir fonctionner indépendamment, sans avoir à adjoindre une machine additionnelle, et sans présupposer que l'utilisateur est assis face à l'ordinateur posé sur une table.

Capacités de calcul limitées

Nous l'avons déjà évoqué dans la partie précédente, les PDA et téléphones portables ont des ressources de calculs et de mémoire limitées. L'interface utilisateur doit donc être la moins gourmande possible en ces termes que possible afin d'offrir une interaction efficace et surtout les ressources nécessaires aux applications.

Faible extensibilité

La dernière caractéristique devant être prise en compte est la faible capacité d'extensibilité des TMC. Les possibilités de connexion et le manque de bibliothèques standards pour les piloter rendent l'intégration de nouveaux composants d'entrées/sorties difficile. En effet, un des défis pour une nouvelle interface est de pouvoir fonctionner sur différentes architectures (logicielles et matérielles).

Vers de nouvelles solutions

Durant cette thèse, nous avons mis au point, en collaboration avec Martin Hachet, deux techniques d'interaction avec les ordinateurs de poche.
La première est une technique de déplacement pouvant s'appliquer aussi bien en 2 qu'en 3 dimensions. Il s'agit d'une interface *tangible* exploitant la caméra de plus en plus souvent embarquée dans de tels ordinateurs. Nous l'avons baptisée TangiMap et nous la présentons et l'évaluons en détail dans le chapitre 8.
La deuxième technique est une technique de sélection plus particulièrement adaptée aux ordinateurs ne disposant pas de dispositif de pointage, en particulier de stylet ou écran tactile comme la majorité des téléphones portables. Notre interface permet à l'utilisateur de sélectionner rapidement un point ou une zone de l'écran à l'aide des touches directionnelles ou du mini joystick disponible. Elle offre une alternative efficace à la simple émulation d'un pointeur souris. Nommée *Jump and Refine* nous la décrivons dans le chapitre 9.

Chapitre 8

Interface tangible basée vidéo : *TangiMap*

Fig. 8.1: *TangiMap : une interface d'interaction tangible.*

8.1 Introduction

Ce chapitre présente la technique TangiMap, une interface d'interaction tangible originale (voir figure 8.1) que nous avons mise au point avec Martin Hachet. Cette technique a fait l'objet de deux publications dans des conférences internationales : une présentation générale à I3D 2005 [HPG05] puis une évaluation présentée à GI 2005 [HPGG05].

Pour élaborer cette technique, nous devons de garder à l'esprit les 4 contraintes fortes des TMC énumérées précédemment. Notre but est de proposer une interface permettant la manipulation de données telles que les scènes 3D tout en assurant une visualisation confortable, en particulier dans le sens où la surface visuelle n'est pas occultée par la main. Nous avons également porté une attention à la légèreté de notre technique, aussi bien d'un point de vue physique qu'en terme de ressources de calcul.

La suite de ce chapitre est organisée comme suit. Dans la section suivante, nous présentons les travaux de recherche précédents dans le domaine de l'interaction adaptée aux TMC. Nous décrivons ensuite notre interface dans la section 8.3, puis dans les sections 8.5 et 8.4 des exemples d'applications 2D et 3D dans lesquels notre interface peut s'avérer utile. Enfin, la section 8.6 décrit une évaluation de l'interface TangiMap dans le cadre d'une tâche de repérage sur une carte 2D.

8.2 Travaux précédents

Différentes techniques ont déjà été proposées pour améliorer l'interaction homme/machine sur les ordinateurs de poche. Certaines reposent sur les capacités d'entrées/sorties (E/S) existantes, d'autres proposent d'utiliser des extensions matérielles en passe de se standardiser. En particulier, certains travaux proposent d'utiliser, pour certaines tâches d'interaction, différents capteurs de position et orientation, mais aussi le flux vidéo d'une caméra embarquée.

8.2.1 Boutons et pointeurs directs

Les applications actuelles pour TMC telles que les agendas, carnets d'adresses, client de courrier électronique, etc. ont légitimé des interfaces d'entrées simples et réduites à quelques boutons : chiffres, fonctions ou directions pour les entrées discrètes, et, sur les PDA, un dispositif de pointage direct, le stylet, associé à un écran tactile permettant des entrées continues. Ces interfaces d'entrées conviennent parfaitement pour de nombreuses tâches simples comme la saisie de chiffres ou la sélection d'un item dans des menus, comme expérimenté dans [KS02]. A partir de ces dispositifs, [GT04] propose d'étendre les recommandations générales en IHM de Shneiderman [Shn97] pour convenir aux ordinateurs de poche. Pour des tâches d'interaction de plus haut niveau, des techniques plus évoluées doivent être utilisées. En particulier, la navigation dans un ensemble d'informations, la petite taille de l'écran impose l'utilisation d'interface utilisateur de zoom (ZUI) telles que celles décrites dans [BH94] ou [IH00]. Une limitation des interfaces disponibles sur les machines de poche concerne leurs degrés de liberté, en particulier pour l'interaction avec des environnements virtuels 3D.

8.2.2 Nouvelles interfaces pour terminaux mobiles

Pour surmonter la limitation des interfaces d'entrée standards sur les ordinateurs de poche, différentes approches ont été proposées durant les dernières années. Pierce et Mahaney [PMA03] présentent une approche annexe opportuniste où les utilisateurs tirent bénéfice des ressources d'E/S disponibles à un moment donné (par exemple une télévision ou un clavier). Dusan *et al.* [DGF03] intègrent un système de reconnaissance vocale pour une interaction multimodale. Un autre exemple est l'utilisation d'un clavier portatif *chording* (clavier disposant de peu de touches et permettant à l'utilisateur de saisir un caractère ou une commande en pressant simultanément plusieurs touches) ou de joystick isométrique [KMRH03].

8.2.3 Capteurs de position et d'orientation

De nombreuses interfaces d'entrée sont basées sur les mouvements de l'ordinateur tenus dans la main. Le *ScrollPad* [FLW04] (cf. figure 8.2) est un PDA appareillé d'une souris optique sur la face arrière et permettant la visualisation de grands documents en le déplaçant sur une table. La technique du *Peephole* [Yee03] est une interface comparable où l'utilisateur

déplace dans l'espace le PDA relié à la mécanique d'une souris par un système de câbles et de poulies. Ce travail est inspiré des investigations plus anciennes de Fitzmaurice [Fit93, FZC93]. Des implémentations récentes [Rek96, HPSH00] de cette approche utilisent des capteurs d'inclinaison et des accéléromètres [RP03]. Le problème intrinsèque de ces techniques est que le point de vue de l'utilisateur sur l'écran change en permanence, et amène souvent à perdre l'angle de visualisation optimal.

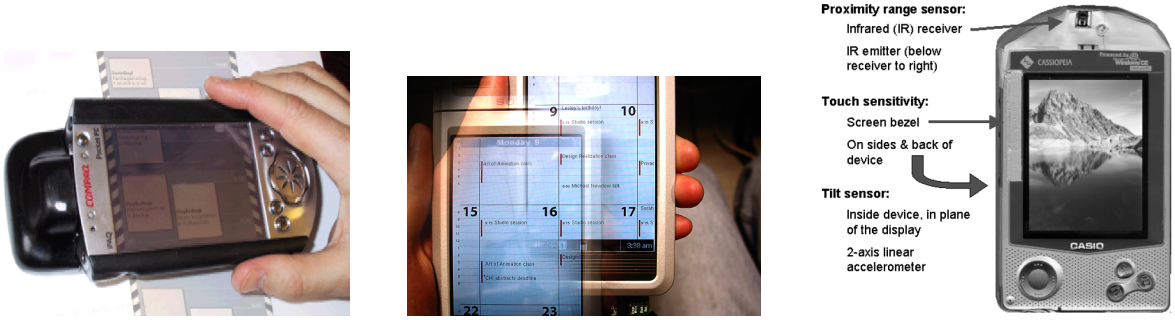


FIG. 8.2: *Techniques du* Scrollpad *[FLW04], du* Peephole *[Yee03] et utilisation de capteurs [RP03].*

8.2.4 Caméras et marqueurs

Les petites caméras embarquées des ordinateurs de poche ont été utilisées dans différentes applications de réalité augmentée. La *NaviCam* de Rekimoto *et al.* [RN95] permet d'obtenir des informations contextuelles lors de la détection de marqueurs spécifiques préalablement disposés dans l'environnement réel. Dans *Invisible Train* Wagner *et al.* [WS03] cette même approche est utilisée pour déterminer la position des marqueurs constitués par une grille de carrés noirs ou blanc (voir figure 8.3 dans le cadre d'une application de réalité augmentée multi-utilisateurs collaborative. Plus récemment, Rohs [Roh04] propose d'utiliser des codes visuels pour différentes tâches d'interaction avec des téléphones portables disposant d'une caméra. Ces approches sont basées sur la détection de marqueurs devant être disposés préalablement dans l'environnement réel. Elles offrent de nombreuses applications intéressantes dans un cadre fixé, par exemple l'affichage d'informations contextuelles à l'approche d'une œuvre dans un musée.

FIG. 8.3: *Techniques basées sur la détection de marqueurs :* Invisible Train *[WS03] et codes visuels de Rohs [Roh04].*

8.2.5 Interaction à deux mains

Notre approche a été également inspirée par les techniques d'interaction bimanuelle (nécessitant les deux mains). Ces techniques ont démontré de nombreux bénéfices [BM86]. En particulier, dans le cadre d'interfaces utilisateurs 3D, l'interaction bimanuelle a été utilisée pour donner à l'utilisateur une référence kinesthésique [BH99]. Le principe est de réaliser des actions avec la main dominante en fonction de la main non dominante servant de référence. Par exemple, Mine [Min98] propose différentes techniques d'interaction pour les environnements virtuels immersifs en utilisant les deux mains pour bénéficier du sens kinesthésique de proprioception (terme de physiologie désignant la capacité du cerveau humain de connaître à tout instant la position du corps dans l'espace).

8.3 Une nouvelle interface basée caméra

8.3.1 Description générale

Inspirés par les forces et faiblesses des approches décrites précédemment, et en prenant en comptes les recommandations que nous avons présentées, nous avons développé une nouvelle interface pour l'interaction avec les ordinateurs de poche.

Afin d'offrir une **bonne visualisation** nous avons opté pour un système dans lequel l'écran n'est jamais occulté par la main de l'utilisateur. Aussi l'utilisateur peut se concentrer sur les données visualisées sans être perturbé par un objet physique. De plus, nous avons particulièrement souhaité que les utilisateurs ne soient pas contraints de déplacer l'ordinateur pour interagir, le laissant ainsi choisir et conserver l'angle de vue optimal sur l'écran. C'est ainsi que dans notre technique les interactions utilisateurs se font en effectuant des mouvements derrière l'ordinateur.

L'usage mobile étant l'intérêt des ordinateurs de poche, nous avons développé un système pouvant être utilisé partout à n'importe quel moment, sans avoir besoin de se connecter à un ordinateur distant. Pour surmonter le problème de la **faible possibilité d'extensibilité**, nous utilisons la caméra qui est de plus en plus souvent intégrée sur ces machines ou qui peut être facilement installée sur un port d'extension mémoire *CompactFlash* ou *SecureDigital*.

Afin de pallier à la **limitation des ressources de calcul** de l'ordinateur de poche, nous avons imaginé une technique efficace. Ainsi, la simple analyse de quelques pixels du flux vidéo nous permet de déterminer instantanément la position 3D de la cible par rapport à l'ordinateur. De par sa rapidité, notre algorithme préserve les ressources essentielles au bon fonctionnement de l'application.

Pour résumer, l'utilisateur interagit avec les données en déplaçant une cible derrière l'écran de l'ordinateur comme illustré dans la figure 8.4. La cible proposée a une taille similaire à celle d'une boite de CD-ROM. La position de la cible est détectée dans l'espace, faisant de notre interface une interface à 3 degrés de liberté. Cette approche tire bénéfice d'un mode d'interaction bimanuelle et du sens kinesthésique décrit plus haut.

FIG. 8.4: *La caméra de l'ordinateur de poche permet l'acquisition des mouvements de la cible déplacée derrière l'écran.*

8.3.2 Mise en œuvre

Des bibliothèques de vision par ordinateur tels qu'OpenCV[1] ou ARtoolkit[2] permettent de suivre en temps réel une cible sur une station de travail. Par exemple, Woods *et al.* [WMB03] utilise ARToolKit pour émuler une souris à 6 degrés de liberté. Cependant, l'utilisation de tels outils sur ordinateurs de poche est difficilement raisonnable, car trop coûteux en temps de calcul. De plus, l'accès à un serveur de calcul par l'utilisation d'une connexion sans fil afin de leur laisser appliquer les algorithmes de vision par ordinateur comme dans [WS03] est relativement critique, car la réponse à l'action de l'utilisateur devient fortement dépendante de la qualité de service offert par le réseau. Il va sans dire qu'une telle approche nécessite donc une infrastructure contraignante difficilement imaginable dans un cadre de mobilité. Par conséquent, nous avons mis au point un algorithme rapide et efficace pouvant être exécuté sur les ordinateurs de poche.

Le suivi 3D d'un simple motif noir sur une cible blanche est une tâche aisée. Cependant, le faible champ de vision offert par les caméras embarquées rend cette technique inefficace, car impliquent que la cible doit être suffisamment éloignée de la caméra pour être entièrement visible pendant son déplacement. De plus, les images du flux vidéo doivent être complètement analysées (c'est à dire parcourues) ce qui entraîne un coût de calcul non négligeable en temps réel.

Afin de favoriser des mouvements larges et de réduire le nombre d'opérations nécessaire au suivi, une autre approche pourrait consister en l'utilisation d'échelles de couleurs. Ainsi, la position (x, y) de la cible pourrait être simplement déduite par la couleur du pixel central

[1] OpenCV : http ://www.intel.com/research/mrl/research/opencv
[2] ARToolKit : http ://www.hitl.washington.edu/artoolkit

de l'image acquise. Malheureusement, la faible qualité des caméras disponibles et la variation éventuelle des conditions lumineuses en mobilité rendent cette approche caduque.

8.3.2.1 Codes couleur RVB

Afin de permettre des mouvements larges, proches ou éloignés de la caméra, sans être trop dépendant des conditions lumineuses, nous avons imaginé un codage à l'aide des couleurs rouge, vert et bleu. La cible proposée est illustrée dans la figure 8.5 et est composée de plusieurs cellules séparées par des bords rouges (couleur choisie au hasard parmi les 3 couleurs de base). La cible est un carré de 12cm×12cm divisé en $8 \times 8 = 64$ cellules. Chaque cellule est elle-même composée de 2 lignes horizontales. La ligne du haut code la coordonnée en abscisse x de la cellule tandis que la coordonnée en ordonnée y est codée dans la ligne du bas. Chacune de ces 2 lignes est divisée en un triplet de cases bleues ou vertes. En assignant la valeur 0 au bleu et 1 au vert, chaque triplet correspond à un code binaire pouvant désigner un entier entre 0 et 8.

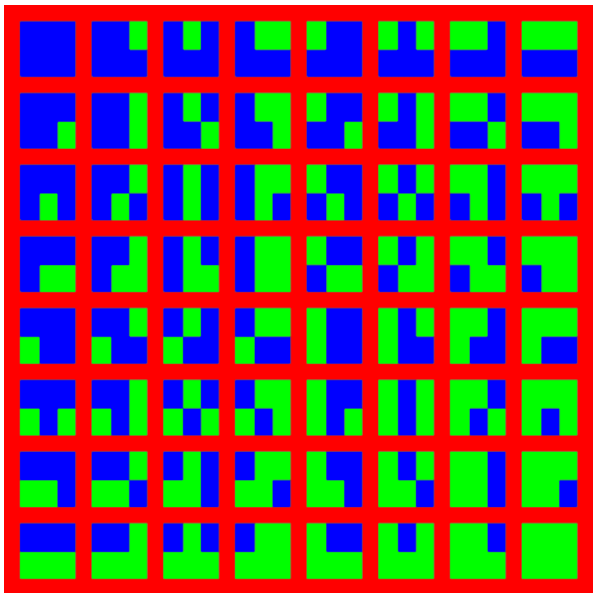
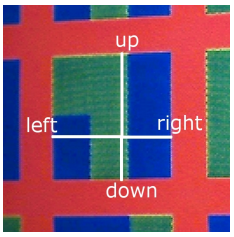


FIG. 8.5: *La cible RGB est composée de cellules décrivant un code binaire.*

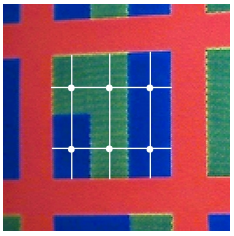
Un pixel dans l'image vidéo acquise est défini par ses trois composantes RGB. Chaque pixel analysé par notre algorithme est classé dans une des trois classes suivantes $ROUGE, VERT, BLEU$. Si les conditions lumineuses sont suffisantes, cette classification peut se faire simplement en fonction de la composante RVB la plus élevée. Sinon, une technique de classification basée sur l'analyse de l'histogramme RVB peut être utilisée, mais celle-ci nécessite une analyse l'intégralité de l'image et donc un temps de traitement plus long.

Dans le cas général, il est très simple (et rapide) de déterminer quel est le point de la cible sur lequel la caméra pointe. Nous décrivons maintenant notre approche. Nous appelons *graine*, le point de référence, initialement situé sur le centre de l'image. Les images sur la droite correspondent à des morceaux d'images acquises par la caméra.

A partir de la graine, nous cherchons les bords de la cellule dans les 4 directions cardinales en suivant une règle simple : *tant que le pixel n'est pas rouge, considérer le pixel suivant.*



Une fois les bords de la cellule identifiés, il est simple de déterminer le code couleur en analysant les six pixels illustrés à droite.



Les codes couleur déterminent la position de la cellule actuellement pointée dans la grille. Dans notre exemple, la première ligne de la cellule possède le code binaire 110, ce qui correspond à la colonne n°6. La seconde ligne est 010, correspondant à la ligne n°3.

La position pointée par le centre de la caméra possède la coordonnée x_{cible} suivante sur la cible :

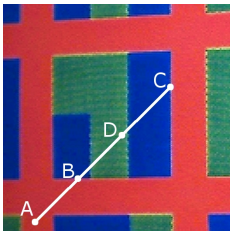
$$x_{cible} = c \times L + gs \times \frac{L}{gd} \tag{8.1}$$

où c est le numéro de colonne, L la largeur d'une colonne sur la cible, gs correspond au nombre de pixels entre *gauche* et *seed*, et gd est le nombre de pixels entre *gauche* et *droite*. La coordonnée y_{cible} est calculée de façon similaire.

La position (x, y) de la cible en fonction de la caméra est donnée directement par x_{cible} et y_{cible}. La coordonnée z est inférée par gd, le nombre de pixels entre *gauche* et *droite*. En effet, plus éloignée est la cible de la caméra, moins la distance $droite - gauche$ est grande.

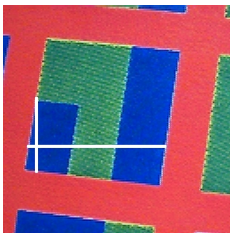
Nous avons décrit la méthode permettant de déterminer la position de la cible quand la graine appartient à une cellule. Quand tel n'est pas le cas, elle doit être déplacée jusqu'à la cellule suivante comme suit.

A est la graine initiale. *Tant que le pixel courant est rouge, se déplacer sur le pixel voisin en haut à droite.* Déterminer B. *Tant que le pixel courant n'est pas rouge, se déplacer sur le pixel suivant en haut à droite.* Déterminer C. La nouvelle graine D est définie par le centre du segment BC.

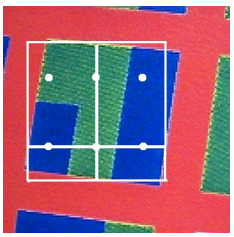


Afin de rendre la technique plus robuste, un dernier test est réalisé avant de chercher les bords de la cellule. Ce dernier test consiste à détecter l'erreur qui pourrait venir d'une inclinaison de la cible.

Les cellules étant carrées, les longueurs $droite - gauche$ et $haut - bas$ sont supposées être égales. Si ce n'est pas le cas, nous sommes dans la situation illustrée à droite.



Dans ce cas particulier, nous déplaçons la graine au milieu du plus long segment entre $droite - gauche$ et $haut - bas$. L'application de cette heuristique permet de gérer les petites inclinaisons, et s'avère suffisante par expérience.



La technique décrite ici fonctionne bien dans différentes conditions lumineuses. Les utilisateurs sont capables de réaliser des déplacements grands et fins de la cible, à différentes distances de la caméra. Ces déplacements sont traités par notre algorithme qui fournit en temps-réel les coordonnées x, y, et z. Ces coordonnées sont des coordonnées absolues dans la mesure où ils ne dépendent pas des coordonnées précédentes. Cependant, lorsque les valeurs détectées sont trop différentes des 3 dernières (ie. leur distance est supérieure à un seuil λ), elles sont considérées comme erronées et ne sont pas prises en compte.

8.3.3 Considérations techniques

Pour les premiers tests, nous utilisons un PDA de type PocketPC Toshiba e800 disposant d'un processeur XSCale ARM cadencé à 400MHz. L'application a été programmée en C et utilise la bibliothèque Microsoft GAPI (Game API) permettant d'offrir un accès à la mémoire graphique direct et donc plus performant. Avec cette machine les coordonnées peuvent être estimées en moins d'un quart de milliseconde. Ces performances permettent d'affirmer que notre approche permet une interaction en temps réel et peut être utilisée sans pénaliser l'application principale. Grâce à son efficacité, notre interface peut être utilisée avec des machines encore moins puissantes comme des téléphones portables.

Nous avons doté notre PDA d'une caméra FlyCam à l'aide du port d'extension Compact Flash. Cette caméra permet d'obtenir un flux d'images de taille 160×120 pixels. Comme expérimenté dans [WS03], cette caméra n'est cependant pas en mesure de délivrer plus de 7 à 8 images par seconde, ce qui constitue le principal goulot d'étranglement. Cependant, nous pensons (et constatons) que de plus en plus de caméras équiperont en standard les ordinateurs de poche et que les flux vidéos seront par conséquent plus facilement exploitables.
D'un point de vue pratique, une fois pliée, la cible peut-être placée dans la pochette du PDA.

8.4 Interaction 3D

Les techniques d'interaction décrites ci-après montrent comment TangiMap peut contribuer au développement d'applications 3D sur ordinateurs de poche. Ces techniques d'interaction sont liées à la manipulation, la navigation, la sélection et le système de contrôle.

8.4.1 Manipulation

Notre interface fournit 3 degrés de liberté pouvant être facilement attachés à différentes opérations telles que la translation ou la rotation d'un objet 3D en fonction du bouton pressé. La correspondance entre les translations de la cible tenue en main et de l'objet manipulé est alors directe. Elle donne à l'utilisateur une sensation physique sur l'objet virtuel manipulé. L'opération de rotation d'un objet est moins directe, car la translation de la cible a pour effet une rotation de l'objet. Cependant, tous les utilisateurs ayant testé TangiMap ont immédiatement compris son fonctionnement sans explication et ce parce que le lien entre son action et son résultat à l'écran fonctionne bien. La correspondance entre le mouvement de la cible et la rotation d'un objet est illustrée dans la figure 8.6.

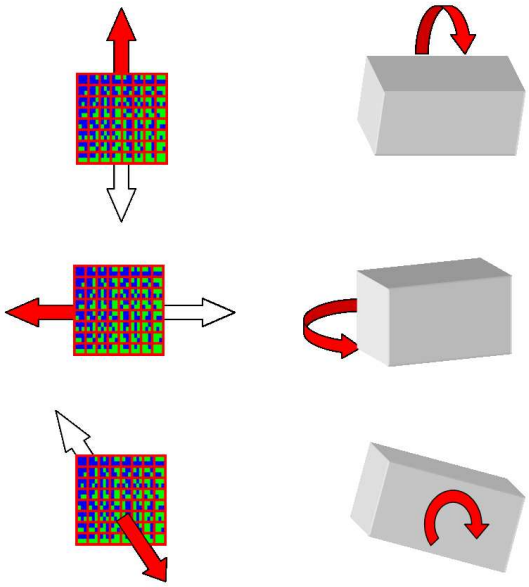


FIG. 8.6: *Correspondance entre les mouvements de la cible et la rotation d'un objet.*

8.4.2 Navigation

Les 3 degrés de liberté fournis par TangiMap peuvent être utilisés pour contrôler les translations de la caméra puis ses rotations. Cependant, nous avons préféré implémenter une technique permettant de naviguer plus efficacement dans une scène 3D en utilisant le degré de liberté *avant-arrière* pour contrôler la vitesse de déplacement dans la scène. Les deux autres degrés de liberté permettent de modifier les angles de lacet et de tangage. Ainsi, en contrôlant ces 3 degrés de liberté en même temps, l'utilisateur peut naviguer facilement dans de grandes scènes 3D : pour aller à droite, il suffit de déplacer la cible à droite, pour aller en haut il faut monter la cible, etc. Un ralentissement se fait en ramenant la cible plus près de la caméra. Une fois encore, la visualisation de la scène n'est plus altérée par le passage de la main tenant le stylet et l'écran peut être orienté confortablement. La figure 8.7 montre un exemple d'une telle navigation 3D sur un MNT.

FIG. 8.7: *Navigation dans un modèle numérique de terrain*

8.4.3 Sélection et système de contrôle

Différentes techniques d'interaction pourraient être imaginées pour sélectionner des objets dans une scène 3D avec TangiMap. Par exemple, une technique pourrait consister à entourer l'objet à l'aide de la cible ou à étendre la sélection rectangulaire 2D dans l'espace 3D. Une autre approche pourrait s'inspirer de la technique utilisée par Encarnação *et al.* [EBSC99] avec leur *translucent sketchpad* basée sur un système de reconnaissance de gestes simples (par exemple, une croix pour supprimer ou un zig-zag pour annuler).

Le contrôle d'applications est un problème réel lorsque l'on s'intéresse aux ordinateurs de poche. En effet, la taille réduite de l'écran rend difficile l'utilisation de grandes barres de menu déroulant telles que celles utilisées sur les ordinateurs de bureau. De nouvelles interfaces utilisateurs doivent donc être imaginées pour permettre un contrôle efficace. Nous proposons d'utiliser une structure de menus hiérarchiques se raffinant en fonction de l'item sélectionné. Avec TangiMap, le niveau de hiérarchie peut être contrôlé par le degré de liberté z tandis que les degrés x et y permettent de naviguer parmi les différents items d'un niveau donné. Par exemple, on pourrait imaginer qu'au premier niveau (correspondant à une position éloignée de la cible) 5 items divisent l'espace : "Fichier", "Editer", etc. En rapprochant la cible de la section "Fichier", 8 nouveaux items apparaissent et ainsi de suite. Après un petit moment, les utilisateurs savent où les items sont situés sur la cible et opèrent alors rapidement sur l'application (par exemple, l'item "Sauvegarder sous" sera dans le haut-gauche de la cible).

8.5 Navigation dans des données 2D

Le développement de TangiMap a été initié par la volonté de visualiser de grandes cartes topographiques ou urbaines sur PDA. Imaginons un chercheur ayant besoin de préparer ses déplacements dans une grande ville qui lui est inconnue pour assister à une conférence. Sur le site de la manifestation, il télécharge sur son ordinateur de poche un plan détaillé indiquant la position de son hôtel et celle de la conférence située à 3 km. Une fois dans la rue, il est nécessaire de visualiser les points de départ et d'arrivée de façon globale afin de pouvoir les situer l'un par rapport à l'autre. A un tel niveau, il n'est cependant plus possible de lire le nom des rues et il est alors nécessaire de zoomer sur le point de départ pour se concentrer sur cette zone là. A ce moment, il faut déplacer la carte jusqu'au point d'arrivée, mais à un tel niveau de rapprochement, on ne sait plus exactement dans quelle direction aller, car la destination n'est plus visible. Par conséquent, il est nécessaire de dé-zoomer afin d'obtenir une vue globale, puis de zoomer et ainsi de suite. Avec un stylet, les opérations de zoom avant et arrière sont généralement réalisées par le biais de boutons dans l'interface graphique et dans cette situation, la tâche devient difficile. Le même type de problème est rencontré avec la plupart des outils en ligne permettant de calculer des itinéraires.

Avec TangiMap, la préparation d'un itinéraire est rendue plus naturel, car on peut établir une correspondance entre la cible et la carte, et avoir ainsi l'impression de tenir toute la carte dans la main. Les opérations de zoom sont réalisées de façon continue en rapprochant ou éloignant naturellement la cible de l'ordinateur.

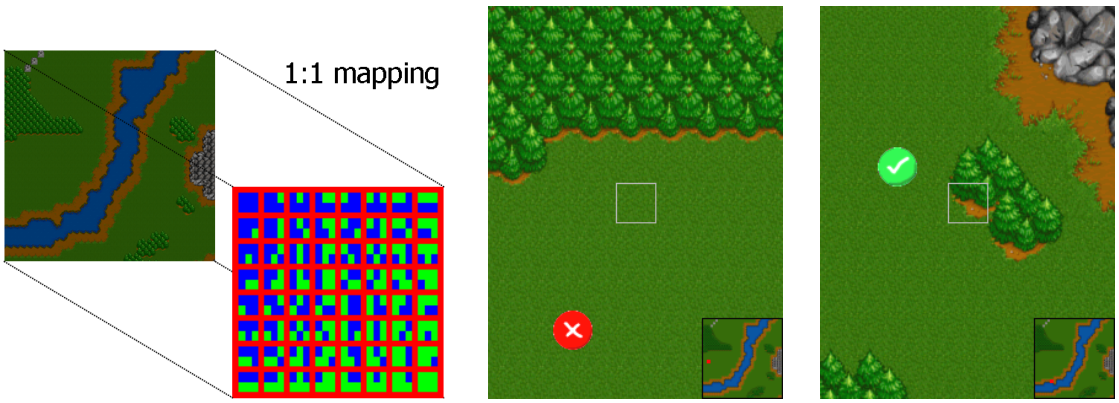
L'utilisation de TangiMap pour la visualisation de grands documents textuels comme des fichiers PDF ou des pages Web comme dans [IH00] n'est pas inappropriée. En effet, la lecture d'un long texte sur un petit écran est une tâche linéaire ne nécessitant qu'un degré de liberté. Par conséquent, les interfaces 1 DOF étendues avec 1 degré de contrôle pour le zoom sont plus adaptées que TangiMap. TangiMap est en revanche plus efficace pour la visualisation de documents 2D telles que des cartes ou photos.

TangiMap est aussi particulièrement bien adapté à la navigation multi-échelle telle que celle utilisée par Bederson *et al.* [BH94] pour leur interface de zoom *Pad++*. Dans ce cadre, la navigation dans les différents axes pourrait être directement contrôlée par les 3 degrés de liberté de TangiMap. De façon similaire, une application de dessin vectoriel par exemple basée sur le standard SVG(*Scalable Vector Graphics* [FJJ03]) peut tirer bénéfice de TangiMap, en particulier lorsqu'elle est couplée à un niveau de détail adaptatif tel que celui décrit par Chang *et al.* [CCW04] et assez similaire au nœud LOD de VRML97. Par exemple, un architecte désirant vérifier sur site la conformité d'un bâtiment avec les plans peut facilement, avec son PDA, zoomer sur telle ou telle partie pour obtenir le niveau de détails souhaité ou au contraire avoir une vue plus globale.

8.6 Evaluation

Nous avons évalué les bénéfices de notre interface par rapport à une approche classique à l'aide du stylet dans le cadre d'une tâche de recherche 2D. Pour ce faire, nous avons mis au point une expérience durant laquelle les sujets doivent trouver des cibles sur une carte. Dans une première étape, nous voulions simplement comparer les différences de performance quand les 2 degrés de liberté horizontal-vertical étaient exploités. En effet, comme les stylets n'ont que 2 degrés de liberté, une technique arbitraire devrait être choisie afin d'émuler le 3ème degré permettant de réaliser des zooms. Pour effectuer les déplacements à l'aide du stylet, nous avons retenu l'approche classique par *drag-and-drop*. Avec TangiMap, les déplacements

sont eux effectués naturellement par les mouvements de la cible et un bouton du PDA est utilisé pour activer ou désactiver le suivi de la cible. Notre hypothèse était que même restreinte à 2 degrés de liberté, les performances de l'utilisateur avec notre interface dépasseraient les performances obtenues à l'aide du stylet.



(a) Correspondance entre la carte visualisée et la cible TangiMap.

(b) Signalée en rouge, la cible n'as pas encore été atteinte, sa position sur la carte est signalée sur la mini-carte en bas à droite.

(c) La cible en vert a été atteinte et la position de la cible suivante à atteindre est maintenant signalée sur la mini-carte.

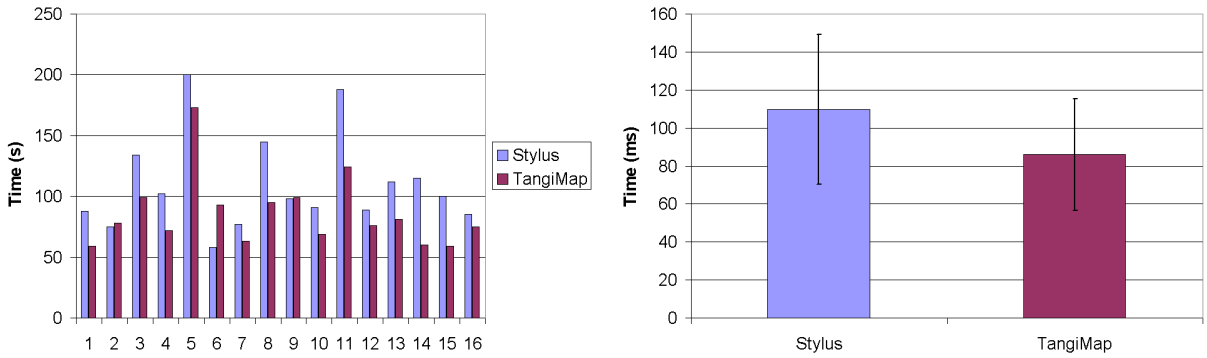
FIG. 8.8: *Copies d'écrans de l'expérience d'évaluation 2D.*

8.6.1 Tâche et procédure

La tâche consiste à chercher des petites cibles qui apparaissent au fur et à mesure sur la carte. Nous avons choisi une carte assez simple mais disposant d'éléments caractéristiques : une rivière, des zones rocheuses et forestières. Comme illustrée dans la figure 8.8, la partie visible de la carte est affichée en plein écran tandis qu'une représentation globale de la carte apparaît dans une petite zone dans le coin en bas à droite, nous l'appelons la mini-carte. Aucune indication sur la position du point de vue n'est donnée dans cette mini-carte. En conséquence, l'utilisateur doit rechercher la cible sur la carte principale et n'a pas à se concentrer sur cette mini-carte. En effet, dans une expérience préalable, la zone affichée était précisée sur la mini-carte et nous avions constaté que les sujets ne regardaient plus la carte, mais se concentraient sur la mini-carte. La tâche était alors plus proche d'une tâche de pointage 2D que d'une tâche de recherche. De plus, l'opération de clic sur la mini-carte avec le stylet pour sauter directement à la position pointée n'était pas autorisée. La téléportation sur une carte n'implique pas de connaissance de l'environnement, ce qui n'est pas en adéquation avec une tâche de recherche. Par exemple, sauter directement à la destination ne donne aucune information sur la route à suivre dans une application de géopositionnement comme celle précédemment décrite.

Pour chaque essai, une cible rouge apparaît de façon aléatoire sur la carte, et le point correspondant est illuminé dans la mini-carte. Les sujets doivent alors atteindre la cible le plus vite possible en la positionnant au centre de l'écran en déplaçant la carte. Une fois la cible touchée, celle-ci devient verte et l'expérience continue avec une autre cible.

16 sujets ont participé à l'expérience menée sur les 2 types d'interface. Toutes ces per-



(a) Temps de réalisation des tâches par les sujets.

(b) Moyennes des temps de réalisation.

FIG. 8.9: *Résultats expérimentaux.*

	Stylet	TangiMap
Temps moyen de réalisation (s)	109.81	85.93
Déviation standard σ	39.37	29.36
	$t(15) = 3.89$, $p = 0.001$	

TAB. 8.1: *Analyse statistique des résultats expérimentaux.*

sonnes, constituées de 10 hommes et 6 femmes, sont des personnes diplômées d'âge variant entre 22 et 35 ans, et aucun d'entre eux n'avait utilisé un PDA plus de 5 fois. La moitié a commencé l'expérience avec le stylet, l'autre avec TangiMap.

Pour chaque interface, les sujets ont réalisé 5 essais d'entraînement puis 15 essais ont été mesurés. 240 enregistrements ont donc été obtenus pour chacune des interfaces. Ces enregistrements contiennent essentiellement le temps mis pour atteindre chaque cible. En revanche, la précision de l'interface, c'est-à-dire le taux d'erreur obtenu pour un positionnement précis, n'a pas été mesurée. Sur ce plan, il est indéniable qu'une interface par pointage direct serait plus précise que TangiMap. Dans l'expérience menée, nous nous sommes donc focalisés sur la rapidité de l'utilisateur pour atteindre les cibles sur la carte ne pouvant être visualisée dans son ensemble.

Une fois l'expérience terminée, les sujets ont rempli un questionnaire d'évaluation.

8.6.2 Résultats

Les résultats de l'étude effectuée sont illustrés par la figure 8.9. Nous avons utilisé la méthode du *t-test* couplé pour effectuer l'analyse statistique des moyennes obtenues. Les résultats de cette analyse sont donnés dans le tableau 8.1. Ils permettent d'affirmer qu'avec TangiMap, l'action de recherche a été significativement plus rapide qu'avec le stylet.

8.6.3 Discussion

Les résultats obtenus peuvent s'expliquer par la structure de notre interface. En effet, en tenant la cible tangible, les utilisateurs bénéficient d'un retour kinesthésique en plus du retour visuel. Par exemple, quand une cible apparaît dans le coin haut-gauche, l'utilisateur

n'a qu'à déplacer directement la cible TangiMap à la position correspondante. Ainsi, la cible est rapidement trouvée. Quand une nouvelle cible apparaît, l'utilisateur déplace avec sa main dominante la cible TangiMap à la nouvelle position en fonction de sa main non dominante.

D'un autre côté, l'interaction à l'aide du stylet s'avère ici moins efficace dans la mesure où l'utilisateur doit déplacer la carte sans savoir exactement où aller et le seul retour est visuel. En conséquence, l'utilisateur doit intégrer les particularités géographiques de la carte induisant donc une charge cognitive plus importante.

Comme expérience informelle, nous avons testé avec quelques sujets la même expérience en substituant le fond de carte par un fond blanc. Avec le stylet, les utilisateurs étaient totalement perdus, car le retour visuel n'était pas suffisant pour permettre le repérage. Avec TangiMap, les cibles étaient facilement détectées du fait de la correspondance carte / cible.

Une technique plus rapide avec le stylet sera de pouvoir cliquer directement sur la destination dans la mini-carte. Cette technique n'a pas été retenue, car le but recherché n'est pas d'effectuer un simple saut, mais de visualiser et de comprendre dans toute leur dimension de grands documents telles que des cartes. En effet, comme nous l'avons expliqué plus haut, un saut direct à une position spécifique ne permet pas d'acquérir une connaissance globale de l'environnement.

Finalement, dans cette évaluation, la carte utilisée n'était pas très grande et une correspondance directe entre la carte et la cible de TangiMap était possible. Pour des documents plus grands, une telle technique n'est plus possible et il convient dès lors d'utiliser une fonction de transfert du premier ordre pour permettre des mouvements relatifs.

8.6.4 Commentaires des sujets

12 des 16 sujets (soit 75%) ont dit préférer utiliser TangiMap que le stylet pour réaliser cette tâche. Ils disent avoir particulièrement apprécié l'aspect continu et direct des mouvements de la cible TangiMap par rapport à l'aspect plus répétitif du stylet. En effet, pour de grands déplacements sur la carte, l'utilisateur doit effectuer de nombreux mouvements de *drag'n'drop* avec le stylet, résultant en des saccades désagréables. En revanche, avec TangiMap, les utilisateurs n'ont qu'à faire un seul mouvement avec la main, résultant en une interaction plus directe.

Une technique consistant à déplacer le stylet à partir du centre de l'écran aurait sans doute permis une interaction plus continue. Cependant, les inconvénients d'une telle technique sont une période d'apprentissage plus longue, une occultation permanente de la surface d'affichage et donc la mauvaise détection des objets recherchés.

Aucun des sujets n'a indiqué que la fatigue était un inconvénient de TangiMap dans la partie de libre expression du questionnaire. Cependant, lorsque nous leur avons posé explicitement la question, 8 des 16 sujets (50%) ont trouvé qu'en effet, TangiMap induisait une fatigue plus rapide que l'interface avec stylet. L'autre moitié a en revanche trouvé que tous les mouvements répétitifs de déplacement du stylet engendraient plus de fatigue. Durant l'expérience, nous avons remarqué que tous les utilisateurs tenaient la cible TangiMap le plus près possible de la caméra. En effet, plus les deux mains sont proches, plus la position semble confortable. Cependant, deux problèmes limitent la proximité entre la cible et la caméra. Le premier est qu'au moins une cible doit être visible dans son entier pour la technique de détection. Les résultats expérimentaux vont donc en faveur d'une réduction de la taille des cellules sur la cible de TangiMap. Le second problème est relatif à la faible qualité d'acquisition de la caméra. En effet, on constate que plus la cible est proche, moins la qualité de l'image capturée est bonne.

Avec une nouvelle version de TangiMap, l'algorithme de détection fonctionne bien à partir d'une distance de 5cm, permettant ainsi une utilisation confortable. L'interface se comporte également correctement lorsque la cible est à la distance du bras tendu. Finalement, nous n'avons constaté qu'un seul sujet qui avait tendance à déplacer le TMC plutôt que la cible.

8.7 Conclusion et travaux futurs

Les applications actuelles pour TMC telles que les agendas, carnets d'adresses, client courrier électronique, etc. ont légitimé l'utilisation de quelques boutons et parfois d'un stylet. Ces interfaces d'entrées sont appropriés pour de nombreuses tâches basiques, cependant, elles sont pas toujours la meilleure solution pour des tâches d'interaction de plus haut niveau requises par des applications plus évoluées telles que des navigateurs des données géographiques quelles soient 2D ou 3D.

Dans ce chapitre, nous avons présenté TangiMap, une nouvelle interface que nous avons mise au point pour interagir avec les TMC équipés d'une caméra. Basée sur la détection via le flux vidéo d'une cible tenue en main par l'utilisateur, TangiMap est une interface bimanuelle à 3 degrés de liberté et bénéficiant du retour kinesthésique. L'utilisateur ayant l'impression de tenir les données en main, TangiMap peut être vue comme une interface tangible. Nous avons mis au point un algorithme très léger permettant une interaction en temps réel en ne mobilisant que quelques cycles CPU.

Nous avons montré comment des applications 2D ou 3D pouvaient tirer profit de TangiMap. L'expérience de repérage 2D menée a montré que TangiMap était plus rapide qu'une approche classique par stylet et que les utilisateurs étaient largement favorables à notre interface. Il pourrait être désormais intéressant de comparer TangiMap avec une interface pour stylet de plus haut niveau adaptée par exemple de l'interface de zoom automatique proposée par Igarashi et Hinckley [IH00] pour naviguer dans de grands documents.

Dans de futurs travaux, nous envisageons d'augmenter le nombre de degrés de liberté proposé par l'interface en prenant en compte la rotation de la cible. Les premières expériences menées ont montré que cette extension était assez facilement réalisable. Nous pensons également pouvoir estimer l'inclinaison de la cible, mais celle-ci ne pourra se faire que sur des degrés relativement faibles.

Outre les applications proposées, nous pensons que cette interface possède un potentiel intéressant pour la visualisation d'information telle que la visualisation de grands graphes. De manière similaire aux applications de visualisation d'images que nous avons présentées, les applications de visualisation d'information pourraient bénéficier de l'interaction bimanuelle et de la référence kinesthésique.

Chapitre 9

Technique de sélection : *Jump and Refine*

Fig. 9.1: *Une technique de sélection accélérée à 2 niveaux.*

9.1 Introduction

Les systèmes interactifs nécessitent des interfaces utilisateurs adaptées pour permettre une interaction efficace entre l'homme et la machine. Sur un ordinateur de bureau, les interfaces WIMP (*Windows, Icons, Menus, Pointing devices*) sont souvent les meilleures interfaces existantes pour interagir avec le système. Ce succès est pour une grande part dû à l'efficacité de la souris comme dispositif de pointage.

Aujourd'hui, les applications interactives se développent rapidement sur des téléphones portables comme celui de la figure 9.1, et de nouvelles interfaces utilisateurs apparaissent. En particulier, les icônes et les menus sont très utilisés, mais le "'P"' du WIMP disparaît. En effet, les contraintes imposées par ces téléphones rendent les dispositifs de pointage tels que la souris ou le stylet assez inutilisables. La manipulation avec une seule main des téléphones a mené au développement de petits dispositifs (*joysticks* ou pavés directionnels) pouvant être utilisés avec le seul pouce.

Du fait de la petite taille des écrans, seuls quelques objets peuvent être affichés sur une même image. Dans ces conditions, les sauts discrets d'un élément à l'autre sont plus adéquats

FIG. 9.2: *Interfaces pour le pouce sur différents téléphones portables.*

que le déplacement continu d'un pointeur comme celui de la souris sur les ordinateurs de bureau. En conséquence, les dispositifs d'entrées discrètes tels que celles générées par les touches directionnelles ou un petit joystick sont le plus utilisés sur ces téléphones pour naviguer dans des interfaces graphiques le plus souvent constituées de listes. Tous les téléphones actuels sont équipés de tels dispositifs qui sont même parfois les seules entrées accessibles directement, comme illustré dans la figure 9.2.

Avec l'augmentation des capacités de calcul, des applications plus complexes apparaissent sur ces téléphones. Comme nous l'avons vu au long de ce mémoire, les applications 3D se développent et nous pouvons facilement imaginer que de telles applications vont devenir de plus en plus populaires sur téléphones mobiles. Cependant, l'utilisation de telles applications nécessite des tâches d'interaction avancées où la seule possibilité consistant à sauter d'un élément à un autre n'est plus suffisante. Il est alors souvent nécessaire de sélectionner un point ou une zone de l'image à l'aide d'une technique de pointage.

Le pointage est une tâche primaire souvent utilisée dans le cadre d'interaction avec des scènes 3D. Par exemple, la technique classique de sélection (*pick*) d'un objet 3D se fait en pointant sa projection sur l'écran. Cette sélection est généralement la première étape avant des tâches plus avancées d'édition ou de manipulation. Une autre technique de base consiste à sélectionner un point d'une surface comme point d'arrivée pour une trajectoire dans une tâche de navigation. Cette technique connue sous le nom de "'*go to*"' est particulièrement intéressante, mais s'avère délicate sur un ordinateur mobile où les entrées discrètes rendent les techniques continues inefficaces. En effet, le contrôle simultané de la direction et de la vitesse réalisé habituellement avec une souris et un clavier n'est pas adapté aux dispositifs actuels des téléphones.

Le développement d'applications géographiques et les technologies de repérage par satellites offrent de nouvelles perspectives pour les ordinateurs mobiles. De telles applications d'aide à la navigation sont apparues ces dernières années et se sont répandues très rapidement. Cependant, toutes ces applications sont souvent 2D ou se limitent à la vue perspective d'un plan, et pourraient tirer profit de la technologie 3D. Imaginons par exemple une application interactive 3D mettant en jeu des environnements urbains sur téléphones mobiles comme illustrée dans la figure 9.1. Les intérêts de telles applications sont facilement détectables. Par exemple, un piéton peut accéder à des informations contextuelles en sélectionnant un point d'intérêt (comme connaître le numéro de téléphone de la pharmacie). Il peut aussi déterminer un itinéraire en naviguant dans une ville en 3D (par exemple, comment se rendre à la pharmacie). Nous pourrions citer ainsi bien d'autres scénarios. Le succès de telles applications n'est possible que si des techniques de sélection sont proposées aux utilisateurs. Outre les applications 3D, bien d'autres applications mobiles pourraient tirer bénéfice de techniques efficaces permettant le pointage d'un point précis de l'écran (par exemple des applications de traitement d'images).

Dans ce cadre, nous avons développé et évalué une nouvelle technique de sélection pour

ordinateurs de poche. La technique que nous présentons dans ce chapitre a été l'objet d'un papier court à la conférence internationale CHI [HPTG07]. Baptisée *Jump and Refine*, notre technique, basée sur les dispositifs d'entrées pour le pouce (joystick ou pavé directionnel), repose sur 2 niveaux de déplacement successif. Dans un premier niveau, le curseur saute rapidement entre les cellules d'un quadrillage de l'écran. Puis, dans un deuxième niveau, le curseur est positionné précisément au sein de la cellule si nécessaire.

Après une rapide revue des outils et techniques de sélection existantes, nous décrivons notre approche dans la section 9.3 ainsi qu'une étude utilisateur que nous avons menée dans le but d'évaluer notre technique.

9.2 Dispositifs et techniques de sélection sur téléphones mobiles

Dans le domaine de l'interaction homme-machine (IHM) pour ordinateurs de bureau, le pointage a été l'objet de nombreux travaux de recherche et considéré comme une opération élémentaire fondamentale pour les tâches de sélection. En particulier, la loi de Fitts [Fit54] est généralement utilisée pour prédire le temps de déplacement pour des trajectoires de pointeur/curseur continues [Mac92]. Afin de minimiser ce temps de déplacement, certains auteurs ont proposé d'accroître la taille du pointeur ou de réduire la distance à la cible. Par exemple, Kabbash et Buxton [KB95] introduisirent les curseurs de zone (*area cursors*) où le pointage se fait par zone plutôt que par la pointe du curseur. Grossman et Balakrishnann [GB05] étendirent cette approche avec le *bubble cursor*. Un exemple d'une technique consistant à réduire la distance est la technique de pointage d'objet (*object pointing*) de Guiard *et al.* [GBBL04] où le curseur saute d'un objet pouvant être sélectionné à un autre. Ces techniques ont cependant été imaginées pour un environnement de travail de bureau où l'utilisateur interagit avec un dispositif de pointage.

Dans le domaine de l'IHM dans un cadre de mobilité, peu de travaux spécifiques se sont encore focalisés sur les tâches de pointage ou de sélection. Ren et Moriya [RM00] ont contribué à augmenter les performances des systèmes basés sur des stylets, mais leur technique requiert l'usage des deux mains de l'utilisateur. Pour surmonter la contrainte de l'usage à deux mains, Karlson *et al.* [KBS05] ont récemment proposé *AppLens* et *LaunchTile*. Cette technique permet de naviguer dans des interfaces utilisateurs (GUI) 2D en utilisant les mouvements du pouce sur l'écran de l'appareil. S'ils sont équipés d'écrans tactiles, les TMC peuvent effectivement bénéficier de cette approche. Cependant, les contraintes anatomiques (en particulier la largeur et la taille du pouce) rendent son usage imprécis. De plus, les écrans tactiles sur de telles machines sont connus pour leur fragilité et les téléphones portables en sont rarement équipés.

Des mini-joysticks isométriques (dits *rate-controlled*) pour le pouce ont été développés pour fournir des mouvements d'entrée continus au système. Par exemple, Silfverberg *et al.* [SMK01] suggèrent l'utilisation d'un joystick isométrique comme dispositif de pointage pour les terminaux mobiles. Un autre exemple est le *NaviPoint* de Kawachiya et Ishikawa [KI98]. Ces dispositifs peuvent aider l'utilisateur dans les tâches de pointage cependant, aucun ordinateur de poche existant n'en est équipé. Une interface de pointage potentielle est la technologie *MobiTouch* [MobiTouch] de Synaptics™, dans laquelle un capteur de pression sous le bloc de touches détecte les mouvements du pouce d'une façon similaire à un *touchpad*.

Notre approche utilise les entrées standards disponibles sur les téléphones portables. Différents travaux de recherche en IHM se sont intéressé également à comment les utili-

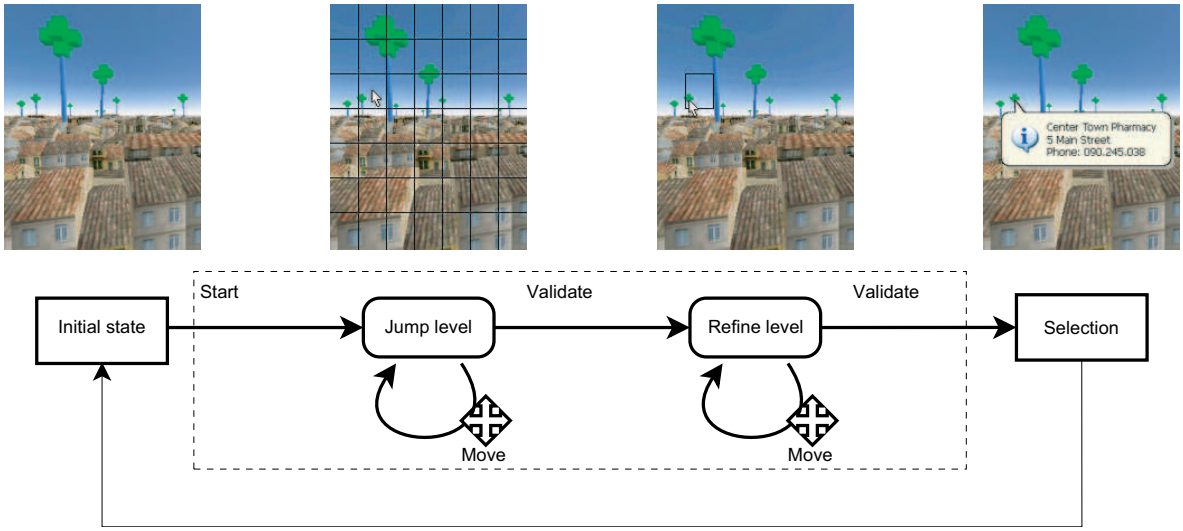


FIG. 9.3: *Schéma de la technique du* Jump and Refine *: le processus de sélection est enclenché au moyen d'une touche dédiée. Dans le niveau* jump*, le curseur est déplacé d'une cellule à l'autre. Un premier appui sur la touche de validation permet de passer au niveau* refine *où le pointeur peut être déplacé précisément au sein de la cellule choisie. Une deuxième validation génère un événement de clic souris à la position déterminée par le pointeur.*

ser pour différentes tâches d'interaction comme la saisie de texte ou la navigation dans des interfaces utilisateurs 2D [AHR04][HN89]. Cependant, à notre connaissance, aucune technique d'interaction n'a été réellement proposée pour améliorer les tâches de sélection sur ces téléphones.

9.3 Jump and refine

Dans cette section, nous présentons l'approche que nous proposons pour atteindre cet objectif. Le but principal étant de minimiser le nombre nécessaire d'appuis sur des touches afin d'améliorer les performances de l'utilisateur comme proposé par les travaux pionniers de Card [CEB87].

Nous avons donc développé une technique dont le but est d'augmenter les performances de l'utilisateur dans les tâches de pointages nécessitant l'utilisation du mini-joystick ou des touches directionnelles. Cette technique n'est pas dédiée à un ou quelques applications spécifiques, mais se veut au contraire comme une technique plus universelle. En conséquence, elle peut être portée sur n'importe quel ordinateur de poche.

L'idée principale de cette technique est d'utiliser deux niveaux de déplacements successifs du curseur. Dans un premier niveau de saut (*jump*), une grille est affichée sur l'écran (voir figure 9.1). Un curseur est positionné dans la cellule centrale, au milieu de l'écran. A partir du dispositif d'entrée directionnel du téléphone, le curseur est déplacé par sauts successifs de cellule en cellule. Cette étape permet un déplacement rapide du curseur. A la suite de l'appui sur une touche de validation, un deuxième niveau de raffinement (*refine*) est activé dans lequel seule la cellule sélectionnée est dessinée. Dans ce second niveau, le curseur peut être précisément déplacé pour atteindre le point désiré au sein de la cellule si nécessaire. Le schéma global de la technique est illustré dans la figure 9.3.

Si la zone à sélectionner est assez grande, le niveau de *jump* est généralement suffisant. C'est le cas si la taille des objets est plus grande que la taille des cellules de la grille. En conséquence, il n'est pas nécessaire de parcourir tous les pixels de l'écran pour atteindre un objet donné. Il suffit de quelques appuis des touches pour de sélectionner la cellule correspondante. Comme aucun raffinement n'est nécessaire, un double appui sur la touche de validation effectuera la sélection. Si un petit objet doit être atteint, ou si une partie précise d'un objet doit être sélectionnée (c'est le cas par exemple dans la technique du "'go to"'), les 2 niveaux de la procédure permettent, dans un premier temps d'atteindre la région d'intérêt, puis dans un second, de positionner finement la position du pointeur. Le pas utilisé pour le déplacement du pointeur dans la 2ème étape, appelé *précision* dans la suite, peut être de 1 ou de quelques pixels pour accélérer la recherche en fonction de la précision souhaitée.

9.3.1 Grilles et résolutions

Du fait de la petite taille des écrans des téléphones portables, la résolution de ceux-ci est généralement faible également (bien qu'elle tende à s'accroître toujours). Par exemple, la résolution classique d'un Smartphone est 176×220. Avec une telle résolution, le nombre optimal de cellules nécessaires pour couvrir l'intégralité d'écran avec un minimum d'appui de touches est assez faible (les performances utilisateurs seront étudiées dans la section suivante). Formellement, ce nombre maximum d'appuis de touches sans le sens horizontal (resp. vertical) est donné par l'équation suivante :

$$Num_{max} = \frac{n}{2} + \frac{resolution}{2.n}, \tag{9.1}$$

avec n le nombre de colonnes (resp. de lignes) :

Pour permettre d'avoir une cellule au milieu de l'écran, n est choisi impair. $\frac{n}{2}$ est ainsi le nombre maximum de cellules du centre au bord de l'écran. Le second terme de l'équation est le nombre maximum de pixels entre le centre de la cellule et le pixel le plus éloigné. Dans cette configuration, il est possible de couvrir l'ensemble des pixels de l'écran. Cependant, en pratique, une précision au pixel près est rarement nécessaire. Une précision de p pixels ($p \in \{1, .., 4\}$) est généralement suffisante pour sélectionner des objets ou pour pointer une direction à suivre, en particulier parce que les objets ne couvrant qu'un seul pixel sont trop petits pour être perçus. En utilisant une précision de p pixels, on augmente d'autant la vitesse de déplacement du pointeur dans le niveau *refine*. En conséquence, le nombre maximum d'appuis de touches est donné par l'équation :

$$Num_{max} = \frac{n}{2} + \frac{resolution}{2.n.precision}. \tag{9.2}$$

Pour une résolution de 176 pixels à l'horizontal et avec une précision de 3 pixels, le nombre maximum d'appuis de touches pour différents nombres de colonnes est donné dans la table 9.1.

9.4 Evaluation

On peut constater que le nombre total d'appui de touches diminue à partir de $n = 3$. A partir de $n = 5$, ce nombre est uniforme jusqu'à 11 colonnes avant d'augmenter. Avec de plus grandes résolutions, la même évolution est constatée avec un décalage du nombre optimal de colonnes sur la droite.

TAB. 9.1: *Nombre maximum de déplacements du curseur dans une direction en fonction d'une résolution de 176 pixels et une précision de 3 pixels.*

Nombre de colonnes	1	3	5	7	9	11	13
Taille des cellules	176	58	35	25	19	16	13
Mouvements au niveau grille	0	1	2	3	4	5	6
Mouvements au niveau pixel	29	9	5	4	3	2	2
Total	29	10	7	7	7	7	8

Pour une résolution et une précision donnée, le nombre optimal de lignes (resp. de colonnes) est atteint quand la dérivée de l'équation 9.2 est nulle, c'est à dire quand :

$$\frac{1}{2} - \frac{1}{n^2}.\frac{resolution}{2.precision} = 0$$

$$\iff n = \sqrt{\frac{resolution}{precision}}.$$

Par exemple, pour un écran de 352 pixels de large, et une précision de 5 pixels, le nombre optimal de colonnes de la grille est 8,39 soit, pour assurer la symétrie, 9 colonnes.

9.4.1 Mise en œuvre

Nous avons implémenté la technique du *Jump and Refine* en C++ sur le système d'exploitation Windows Mobile 2003 pour Smartphone. Techniquement, une touche de raccourci (*hotkey*) est enregistrée pour réveiller notre application qui agit donc comme un service système. De cette façon, notre schéma de sélection est transparent pour l'application en cours d'utilisation par l'utilisateur. Comme le Smartphone ne supporte pas l'affichage d'un curseur de façon native, notre service en affiche un à l'aide de la bibliothèque Game API.

Une fois activée par la touche de raccourci (par exemple la touche d'enregistrement), l'état de l'écran est sauvegardé et la grille est affichée à l'aide de GAPI Les touches de déplacement (éventuellement issues du joystick) sont alors interceptées pour déplacer le curseur sur la grille. Lorsque l'utilisateur valide la position du curseur dans la deuxième étape, l'état de l'écran est restauré et un évènement souris est simulé à l'aide de la fonction `mouse_event()` de l'API WinCE. Malheureusement, comme de tels évènements ne sont pas sensés se produire sur une telle machine (du moins pour l'instant), les applications Smartphone standards ne sont pas attentives aux évènements souris classiques. Nous avons donc développé quelques applications de test telles que celle utilisée dans le cadre de l'évaluation. Dans celle-ci, nous utilisons un ensemble de fonctions fournies par notre application service pour paramétrer la couleur et la taille de la grille.

Dans le monde des ordinateurs de poche, de nombreuses applications ont été développées pour les PDA de type PocketPC disposant d'un stylet. Dans la mesure où les PocketPC sont basés sur Windows Mobile, les applications PocketPC peuvent être directement portées sur Smartphone et ainsi bénéficier de la technique du *Jump and Refine*.

Nous avons souhaité évaluer l'efficacité de l'approche *Jump and Refine* pour la sélection de différents objets dans un environnement 3D. En particulier, nous avons voulu étudier l'impact de la taille de la grille sur les performances des utilisateurs dans une telle tâche. L'expérience

menée a donc consisté à demander à sélectionner des bâtiments sur l'image d'une scène urbaine en 3D à l'aide de notre interface et en utilisant différents paramètres. La description technique de l'expérience est décrite ci-après (participants, environnement technique et tâche). S'en suit une analyse des résultats obtenus ainsi qu'une discussion sur les performances et la satisfaction des utilisateurs.

9.4.2 Participants

L'expérience a été menée sur 20 personnes droitières familières avec l'utilisation d'un tel téléphone portable. Le groupe était constitué de 15 hommes et 5 femmes d'âge allant de 23 à 36 ans. Tous les sujets utilisèrent leur main droite pour tenir le téléphone portable. Avant d'effectuer le test, les participants se sont familiarisés avec le téléphone portable utilisé ainsi qu'avec la technique du *Jump and Refine*.

Une fois l'expérience effectuée, les sujets ont rempli un formulaire de compréhension ainsi qu'un questionnaire de satisfaction. L'étude des résultats obtenus indique que tous les sujets ont bien compris le fonctionnement de l'interface et qu'ils n'ont pas eu de difficulté particulière pour réaliser la tâche. Les résultats du questionnaire sont étudiés plus loin.

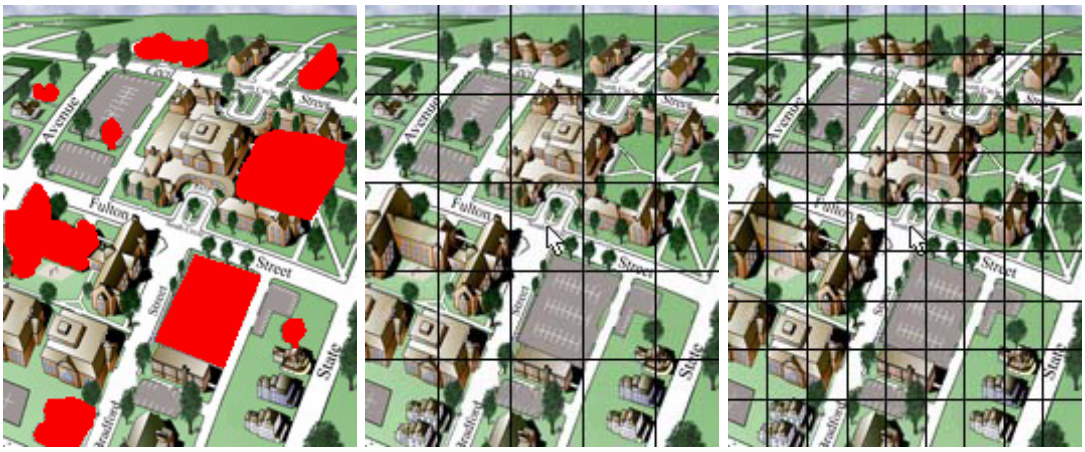
9.4.3 Environnement technique

Le téléphone portable utilisé dans notre expérience était de type HTC Voyager SPV E200, disposant d'un écran de résolution 176×220. Ce téléphone est équipé d'un mini-joystick pour le pouce tel qu'illustré dans la figure 9.1. Le processus de validation avec ce téléphone se fait généralement par pression sur le centre du joystick. Cependant, nous avons constaté que cela induisait souvent des confusions entre le déplacement du curseur et la validation. Aussi, nous avons utilisé une touche voisine comme touche de validation (en l'occurrence la touche de prise de ligne).

9.4.4 Tâche

La tâche proposée consiste à sélectionner des immeubles dans une ville affichée en 3D (voir figure 9.4). Durant l'expérience, chaque sujet doit réaliser la tâche de sélection aussi vite que possible avec 6 grilles différentes pour le niveau de saut : *Grid0*, *Grid3*, *Grid5*, *Grid7*, *Grid9* et *Grid11*. Ces grilles sont respectivement constituées de 1, 3, 5, 7, 9 et 11 lignes et colonnes. Pour chaque grille, la tâche consiste à sélectionner 9 cibles différentes. 3 de ces cibles sont considérées comme petites (environ 10 pixels de large), 3 comme moyennes (environ 20 pixels) et 3 comme grosses (environ 40 pixels). La précision des mouvements du curseur tel que défini dans la section 9.3.1 est fixée à 3 pixels. L'ordre des grilles ainsi que les cibles sont choisis de façon aléatoire.

Après une opération de sélection, le curseur est repositionné au centre de la cellule et une nouvelle cible apparaît. Pour chacune des cibles, le temps de sélection est mesuré, sauf si la cible a été manquée (nous avons cependant constaté que le taux d'erreur était proche de zéro). La moyenne des temps de sélection pour une grille donnée est considérée comme étant le score de l'utilisateur. Nous obtenons donc un score par sujet pour chaque grille.



Grid0 *Grid5* *Grid9*

FIG. 9.4: *Trois exemples de grilles. Toutes les cibles sont affichées dans* Grid0.

9.4.5 Résultats

9.4.5.1 Performances

Afin de comparer ces scores, nous avons effectué une analyse de la variance simple (outil statistique permettant de comparer plusieurs échantillons de données, en anglais, *One-way ANOVA*) dans les mesures répétées. Les statistiques descriptives sont présentées dans la figure 9.5 et la table 9.2.

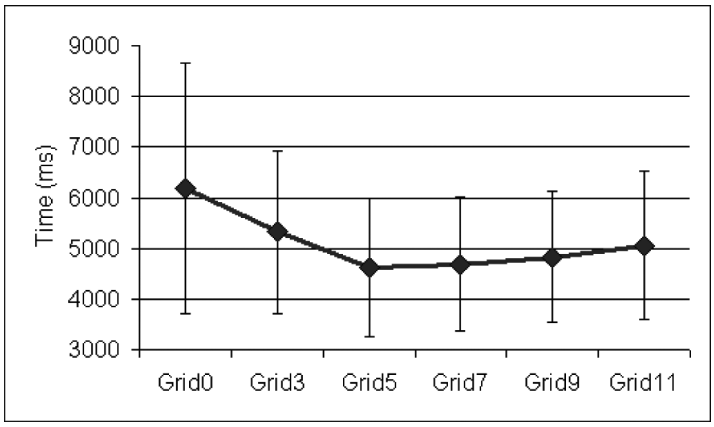


FIG. 9.5: *Performances obtenues pour différentes tailles de grille.*

L'analyse de la variance montre un effet de la taille de la grille ($F(15,5) = 6.78; p < 0.05$). Cette différence significative dans l'analyse de la variance seule montre que les moyennes ne sont pas toutes égales. Nous utilisons le *t-test* de Student [PFTV92] afin d'identifier quelles paires de moyennes sont significativement différentes. Nous ne détaillons ici que les résultats significatifs. Les résultats de ces comparaisons montrent que *Grid0* demande significativement plus de temps que les autres grilles ($(t(19) = 2.46; p < 0.05)$ pour *Grid3*, $(t(19) = 4.02; p < 0.05)$ pour *Grid5*, $(t(19) = 3.51; p < 0.05)$ pour *Grid7*, $(t(19) = 2.84; p < 0.05)$ pour *Grid9*, et $(t(19) = 2.54; p < 0.05)$ pour *Grid11*). De plus nous remarquons que les sujets sont

TAB. 9.2: *Statistiques descriptives pour chaque taille de grille.*

	Temps moyen (ms)	Déviation standard
Grid0	6194.10	2469.02
Grid3	5323.19	1607.97
Grid5	4618.73	1356.60
Grid7	4688.37	1315.65
Grid9	4827.64	1296.07
Grid11	5060.58	1459.99

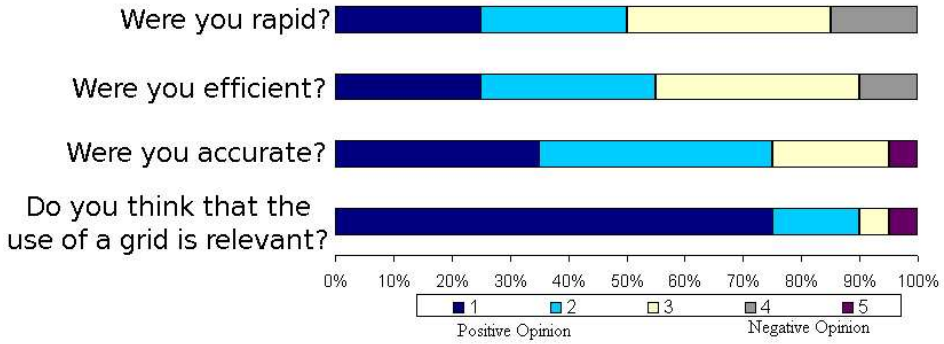


FIG. 9.6: *Résultats de l'étude de satisfaction des utilisateurs pour la technique* Jump and Refine.

plus rapides avec *Grid5* et*Grid7* qu'avec *Grid3* ($(t(19) = 2.83; p < 0.05)$ pour *Grid5* et $(t(19) = 2.19; p < 0.05)$ pour *Grid7*). Il n'existe pas de différence significative entre les grilles 5 à 11. Cependant, les statistiques descriptives montrent que les meilleures performances sont atteintes avec les grilles *Grid5* et *Grid7*.

9.4.5.2 Satisfaction

Pour compléter notre analyse, nous avons mené une étude qualitative sur la satisfaction des utilisateurs. Après l'expérience, nous avons demandé à chaque sujet de répondre à un questionnaire sur la technique *Jump and Refine*. Chaque question pouvant donner lieu à une réponse sous forme de note sur une échelle allant de 1 (excellent) à 5 (très négatif). Les questions posées et les réponses obtenues sont résumées dans la figure 9.6.

Grâce aux réponses obtenues, nous sommes en mesure de conclure que les sujets ont été satisfaits par la technique du *Jump and Refine*et qu'ils l'ont trouvé efficace pour effectuer une tâche de sélection sur une image 3D, seuls 10% ayant exprimé une opinion vraiment négative.

A la question *"Avec quelle grille vous êtes-vous senti le plus efficace"*, 45% ont répondu *Grid7*, 20% *Grid5*, 15% *Grid3*, 10% *Grid11*, 5% *Grid9* et 5% *Grid0*. Ces résultats subjectifs sont cohérents avec les performances mesurées où les meilleurs temps ont été obtenus avec *Grid5* et *Grid7*.

9.5 Discussion et conclusion

Nous avons présenté une technique de sélection efficace en remplacement d'un dispositif

de pointage continu. L'étude menée a montré que la technique proposée est bien assimilée et acceptée par les utilisateurs. Aucun n'a exprimé de difficulté pour intégrer et réaliser la tâche demandée. Les mesures statistiques quantitatives confirment que l'approche par *Jump and Refine* est plus rapide que le déplacement simple d'un curseur sans l'étape de *saut* dans le cadre d'une tache de sélection de cible. Ces résultats vont dans le sens de l'étude de satisfaction qualitative des utilisateurs.

Les résultats de l'étude permettent de lier les temps de réalisation expérimentaux avec le nombre théorique d'appuis de touches requis que nous avons formalisé dans la section 9.3.1. En effet, les meilleurs scores sont obtenus expérimentalement pour les grilles où un minimum de touches est théoriquement nécessaire. Une fois encore, ces résultats sont à mettre en relation avec les préférences des utilisateurs. Des expériences supplémentaires pourraient être menées avec des résolutions d'écran et des précisions de déplacement du curseur différentes afin de vérifier que la grille calculée comme optimale donne bien les meilleurs résultats expérimentaux. La taille de la grille pourrait alors être automatiquement choisie dans une application en fonction de la résolution écran et la précision souhaitée.

Dans l'expérience, nous avons utilisé différentes tailles représentatives de cibles. Il serait intéressant de comparer les temps obtenus en fonction des tailles en augmentant le nombre de cibles. Un autre travail pourrait consister à chercher une loi de prédiction en fonction de la taille de la grille, de la taille de la cible et de la distance initiale à celle-ci.

Dans la mesure où aucun périphérique d'entrée particulier n'est nécessaire, hormis les touches directionnelles (éventuellement liées par un joystick) toujours présentes, la technique ici décrite peut s'appliquer à n'importe quel type de téléphone ou assistant personnel. Les applications nécessitant des tâches de pointage pourraient alors être utilisées plus efficacement même sans dispositif de pointage externe. Une telle technique peut contribuer au développement des applications sur téléphones mobiles, en particulier des applications interactives 3D impliquant des tâches de sélection.

Enfin, la technique du *Jump and Refine* n'est pas restreinte à une utilisation sur téléphones portables, mais pourrait être utilisée dans divers autres domaines. Par exemple, la majorité des télécommandes sont équipées de touches directionnelles. Le *Jump and Refine* pourrait ainsi être utilisé pour des tâches de sélection avancées dans le cadre d'une télévision interactive.

Conclusion générale

Cette thèse, articulée en trois parties, s'est attachée à présenter une chaîne de techniques permettant de créer des modèles de terrains à partir de cartes topographiques, de les visualiser sur des TMC et d'interagir avec eux sur ces terminaux.

Contributions

Les contributions principales de cette thèse sont résumées dans la figure 9.7.

La première partie, consacrée à l'analyse de cartes topographiques en vue de créer des MNT de façon semi-automatique, a permis de mettre en évidence la difficulté de cette tâche. Nous avons cependant proposé une chaîne de traitement des courbes de niveau en trois étapes : segmentation, reconstruction et interpolation. La difficulté principale de ce traitement tient essentiellement dans la complexité d'extraire efficacement les courbes de niveau sur des cartes aux caractéristiques diverses. Dans ce cadre, avons proposé une technique robuste pour reconstruire automatiquement les morceaux de courbes de niveau. Fondée sur la construction d'un champ d'orientation induit par les courbes de niveau vectorisées, cette technique recherche le meilleur appariement d'extrémités de courbes possible. L'utilisation du champ d'orientation permet de reconstruire de façon naturelle un morceau de courbe au lieu d'un simple segment de droite. L'étude de l'interpolation de MNT à partir de courbes de niveau à permis de mettre en évidence les caractéristiques des différentes techniques existantes. La méthode d'interpolation hiérarchique à base de FBR permet de reconstruire efficacement des MNA à partir d'échantillons épars ou de courbes de niveau échantillonnées. Les MNA obtenus reconstruisent les sommets et les puits par un modèle lisse. Enfin, pour mettre en œuvre l'ensemble des algorithmes liés à l'analyse de cartes topographique et la reconstruction de MNT, nous avons développé un logiciel baptisé `AutoDEM`. Mis à la disposition du public et déjà utilisé par différents organismes, ce logiciel a pour but de proposer une plate-forme dédiée à l'expérimentation et à la mise en œuvre pratique des techniques existantes ou à venir.

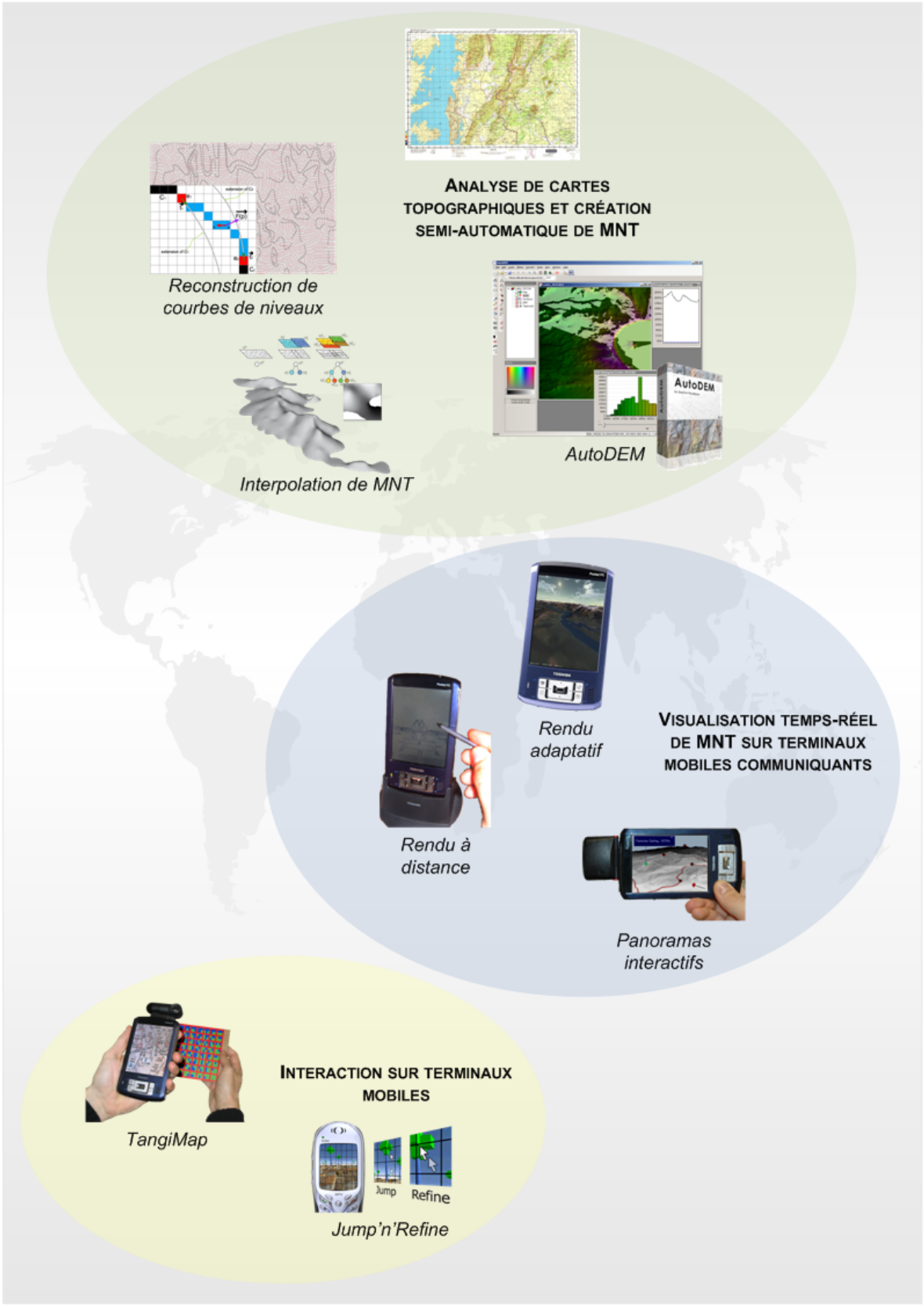


FIG. 9.7: *Principales contributions de cette thèse.*

Dans une deuxième partie, nous avons présenté trois techniques permettant d'offrir à un utilisateur de TMC une visualisation interactive de vastes modèles terrains. Dans un premier temps, après avoir constaté l'impossibilité d'utiliser les techniques existantes sur des TMC aux capacités trop limitées, nous avons cherché à proposer un algorithme permettant d'effectuer le rendu interactif temps réel de larges scènes géographiques. La technique client / serveur proposée s'appuie sur le concept d'adaptativité. Les adaptations en occupation mémoire grâce à un algorithme de pavage du terrain, et en ressources de calculs mobilisées pour dessiner la scène reposant sur une technique de niveaux de détails permettent de garantir un rendu interactif. Une analyse des résultats obtenus a validé la nécessité et l'efficacité d'une telle approche adaptative. A noter que ces travaux de rendu 3D temps réel sur TMC nous ont également donné l'opportunité de développer la bibliothèque `GLUT|ES` permettant de simplifier la création d'applications 3D. Nous avons également contribué à l'évolution de la plate-forme `Elkano` spécialisée dans la mise au point de solutions visualisation de scènes 3D interactives distribuées sur des machines hétérogènes. Les deux autres techniques proposées dans cette partie sont établies sur un rendu à distance. La première, s'adaptant à un milieu desservi par un réseau haut-débit tel qu'un musée ou un site industriel, consiste à utiliser le TMC comme un visualisateur d'une scène calculée à distance. Afin de réduire la quantité de données transitant et d'offrir un taux de rafraîchissement suffisant, une technique de rendu simplifié, mais expressif en noir et blanc est utilisée lors des phases d'interaction. Une fois celles-ci terminées, une image de haute qualité est envoyée au client. La deuxième technique, adaptée à un randonneur ou un touriste dans un site montagneux, est un service de repérage instantané. Par le biais d'un panorama calculé à distance, l'utilisateur se voit délivrer une scène 3D interactive représentant le paysage qui l'entoure ainsi que des informations contextuelles pertinentes.

Enfin, dans la troisième partie, deux techniques d'interaction proposées ont été décrites. Notre but était de proposer des techniques adaptées aux applications de visualisation de données géographiques. La première, reposant sur la détection d'une cible par analyse du flux vidéo, a montré de véritables potentiels dans le cadre d'applications variées. Les trois degrés de liberté offerts par l'interface tangible et bi-manuelle TangiMap peuvent être mis à profit dans de nombreuses applications 2D ou 3D. Nous avons en particulier montré son efficacité pour des tâches de visualisation de grands documents tels que des cartes ou des plans. La deuxième technique, dite *Jump and Refine* a permis de formaliser et de valider expérimentalement une technique simple de sélection en deux étapes : une sélection grossière dans un premier temps puis une sélection précise dans un second. Cette technique, destinée aux TM ne disposant pas de dispositif de pointage direct ou d'entrée continue, permet une évolution des applications disponibles sur ces plates-formes.

Perspectives

Dans le cadre de la reconstruction de MNT, nous pensons que de futurs travaux devront s'attacher à proposer des techniques supervisées plus robustes et spécifiques à l'extraction des courbes de niveau. Dans le cadre d'applications pratiques, il est important d'insister sur la nécessité de travailler sur une plate-forme logicielle spécifique. Le logiciel `AutoDEM` que nous avons développé devra s'attacher à offrir des outils d'aide aux traitements mis en œuvre, par exemple par le biais de modèles ou à l'aide de boites à outils "progressives" s'adaptant à l'étape en cours. Une application de conversion de cartes topographiques en MNT à grande échelle

doit également pouvoir traiter efficacement des cartes de très grandes tailles. Il sera donc important d'étudier des méthodes d'accès aux données et de traitement d'images adaptées à ces grands documents.

La visualisation de données géographiques sur TMC devrait connaître dans les prochaines années un essor à la mesure de son évolution sur les stations de travail. Notre approche de rendu de terrain 3D sur des dispositifs à faibles ressources a montré, quelle que soit l'évolution inévitable à venir des capacités de calcul de ces dispositifs, l'importance d'une technique adaptative, permettant de tirer au mieux parti des propriétés d'un parc de terminaux hétérogènes. Il serait également important de pouvoir proposer, toujours d'une manière adaptative, la visualisation de données supplémentaires telles que les forêts, villes, toponymes, etc. Les applications de repérage augmenté sont également promues à une utilisation grand public dans un avenir très proche. Le service de panorama que nous avons esquissé est une base qui pourra permettre de déterminer techniques de rendu NPR et les informations les plus pertinentes pour une utilisation réelle.

En ce qui concerne l'interaction sur TMC, nous pensons que, de par l'immensité, du marché de nombreuses innovations techniques sont encore à venir. Nous croyons au potentiel d'une technique comme TangiMap, qui présente une alternative intéressante aux boutons et stylets et pourrait se voir augmenter de deux degrés de liberté supplémentaires en détectant sa rotation et son inclinaison.

Pour conclure, cette thèse nous a permis de découvrir des domaines de recherches très différents : géomatique et SIG, segmentation d'image, analyse de document, interpolation de surface, rendu de terrain temps réel, applications distribuées et interface homme-machine. Dans chacune de ces thématiques, nous sommes parvenus à proposer des techniques nouvelles et avons contribué aux développements d'outils adaptés. La découverte du monde de la recherche conjuguée aux compétences théoriques et pratiques acquises durant ces années me permettra sans aucun doute d'appréhender plus aisément des thématiques nouvelles dans le cadre de mon futur parcours professionnel.

Annexes

La bibliothèque GLUT—ES

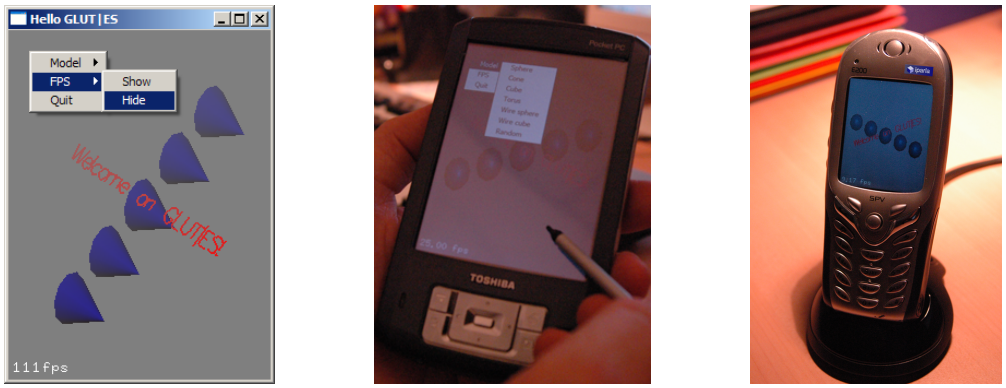


FIG. 9.8: *Application d'exemple fournie avec* ***GLUT|ES*** *et fonctionnant sur PC, PocketPC et SmartPhone*

Durant nos travaux sur la visualisation de terrains, j'ai développé la bibliothèque `GLUT|ES`. Cette bibliothèque est une adaptation *OpenSource* de la bibliothèque `GLUT` (OpenGL Utility Toolkit, mise au point par Mark Kilgard pour son livre OpenGL 'RedBook') pour les ordinateurs embarqués. Celle-ci est pour l'instant compatible avec les systèmes de type Windows Mobile ou Windows PC.

`GLUT` est une API multi plate-forme permettant au développeur de créer et de gérer facilement une fenêtre de rendu OpenGL, de gérer les interactions avec l'utilisateur (clavier, souris ou stylet...), de créer des menus, etc. `GLUT|ES` est basée sur une implémentation OpenSource de `GLUT` appelée `freeglut` initiée par Pawel W. Olszta.

`GLUT|ES` fonctionne donc sur des terminaux de type PocketPC, SmartPhone et PC. Un port Symbian est prévu. `GLUT|ES` est basée sur la bibliothèque de rendu 3D dédiée à ces appareils : OpenGL ES. Cette bibliothèque est une API de bas niveau définissant un ensemble de fonctions permettant de tracer des primitives géométriques 3D à l'écran. OpenGL ES est une version allégée d'OpenGL standardisée par le consortium Khronos et ayant pour but le rendu de scènes 3D de manière logicielle ou matérielle sur des machines à ressources (vitesse, mémoire, batterie, taille d'écran) limitées.

GLUT|ES apporte tout ce qu'apporte originalement GLUT pour les développeurs d'applications 3D sur des stations de travail, à savoir une bibliothèque de fonctions de haut niveau permettant de créer très rapidement des applications 3D munies d'une interface utilisateur. Le principal avantage est la portabilité : l'intérêt de GLUT en général est son rôle d'interface entre le système d'exploitation et l'application (voir figure 9.9). Elle offre un ensemble de fonctions permettant de masquer au développeur le système de fenêtrage sous-jacent ainsi que la gestion des évènements. L'application peut donc être compilée, de manière transparente, sur les différentes plates-formes compatibles avec GLUT|ES (voir figure 9.8).

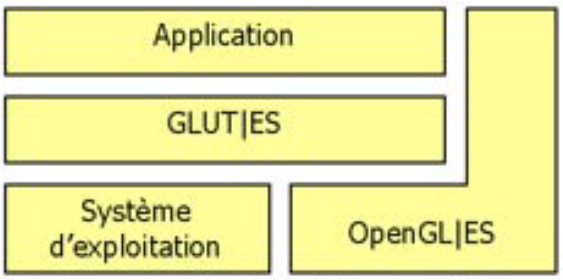


FIG. 9.9: *Positionnement de la bibliothèque GLUT|ES.*

GLUT|ES est distribuée par le biais de SourceForge [GLUTES] et déjà a fait l'objet de plus de 4000 de téléchargements depuis sa mise en ligne début 2005.

A noter que la mise au point de GLUT|ES s'est en partie effectuée grâce au support matériel de la société *Intel*. En effet, souhaitant développer des applications 3D pour PocketPC, un ingénieur d'Intel nous a contactés afin que GLUT|ES supporte l'implémentation OpenGL ES du processeur graphique Intel 2700G. La société Intel nous a alors prêté un PocketPC DELL Axim X51v disposant de ce processeur.

Bibliographie

[AB89] C. Arcelli and G. S. D. Baja. A one-pass two-operation process to detect the skeletal pixels on the 4-distance transform. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 11(4) :411–414, 1989.

[ABE98] N. Amenta, M. Bern, and D. Eppstein. The crust and the β-skeleton : Combinatorial curve reconstruction. *Graphical models and image processing*, 60(2) :125–135, 1998.

[AH05] A. Asirvatham and H. Hoppe. *GPU Gems II*, chapter Terrain rendering using GPU-based geometry clipmaps, pages 27–44. Addison-Wesley, 2005.

[AHR04] R. S. Amant, T. E. Horton, and F. E. Ritter. Model-based evaluation of cell phone menu interaction. In *CHI '04 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 343–350. ACM Press, 2004.

[AMH02] T. Akenine-Moller and E. Haines. *Real-Time Rendering*. A.K. Peters Ltd., 2nd edition, 2002.

[Arr98] P. Arrighi. Reconstruction 3d de cartes numérisées. Master's thesis, Université Montpellier II, 1998.

[AS99] P. Arrighi and P. Soille. From scanned topographic maps to digital elevation models. In *Proceedings of Geovision'99*, 1999.

[Att95] D. Attali. *Squelette et Graphes de Voronoi 2D et 3D*. PhD thesis, Université Joseph Fourier, 1995.

[Aub96] P. Aubry. Analyse et modélisation des systèmes biologiques. Master's thesis, Université Claude Bernard - Lyon 1, 1996.

[Aub03] O. Aubault. *Visualisation Interactive de Scènes Vastes et Complexes à travers un Réseau*. PhD thesis, France Télécom Recherche et Développement, 2003.

[BBF03] A. Bessaid, H. Bechar, and K. Fellah. Image analysis and pattern recognitionas tools in map interpretation. *Electronic Journal Technical Acoustics*, 15, 2003.

[BC05] S. Burigat and L. Chittaro. Location-aware visualization of VRML models in GPS-based mobile guides. In *Web3D '05 : Proceedings of the tenth international conference on 3D Web technology*, pages 57–64. ACM Press, 2005.

[BH94] B. B. Bederson and J. D. Hollan. Pad++ : A zooming graphical interface for exploring alternate interface physics. In *Proceedings of the 7th Annual ACM*

Symposium on User Interface Software and Technology (UIST'94), pages 17–26, 1994.

[BH99] R. Balakrishnan and K. Hinckley. The role of kinesthetic reference frames in two-handed input performance. In *Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 171–178. ACM Press, 1999.

[Blo00] J. Blow. Terrain rendering research for games. Course on Games Research : The Science of Interactive Entertainment at SIGGRAPH, 2000.

[BM86] W. Buxton and B. Myers. A study in two-handed input. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 321–326. ACM Press, 1986.

[Bri74] I. Briggs. Machine contouring using minimum curvature. *Geophysics*, 39 :39–48, 1974.

[Bur86] P. Burrough. *Principles of geographical information systems for land resources assessment.* Clarendon Press, 1986.

[Car88] G. Carrara. Drainage and divide networks derived from high-fidelity digital terrain models. In C. C. et al. Eds., editor, *Quantitative Analysis of Mineral and Energy Resources*, pages 581–597. D. Reidel Pub. Co., 1988.

[CCW04] Y.-H. Chang, T.-R. Chuang, and H.-C. Wang. Adaptive Level-of-detail in SVG. In *SVG Open 2004 : 3nd Annual Conference on Scalable Vector Graphics*, 2004.

[CEB87] S. K. Card, W. K. English, and B. J. Burr. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys, for text selection on a CRT. *Human-computer interaction : a multidisciplinary approach*, pages 386–392, 1987.

[CFCK06] A. Chessel, R. Fablet, F. Cao, and C. Kervrann. Orientation interpolation and applications. In *Proceedings of IEEE International Confernece on Image Processing : ICIP'06*, 2006.

[CGG+03a] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno. BDAM – batched dynamic adaptive meshes for high performance terrain visualization. *Computer Graphics Forum*, 22(3) :505–514, 2003.

[CGG+03b] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno. Interactive out-of-core visualization of very large landscapes on commodity graphics platforms. In *International Conference on Virtual Storytelling*, volume 2897 of *Lecture Notes in Computer Science*, pages 21–29. Springer Berlin / Heidelberg, 2003.

[Chi03] J. Childs. Development of a two-level iterative computational method for solution of the franklin approximation algorithm for the interpolation of large contour line data sets. Master's thesis, Rensselaer Polytechnic Institute, 2003.

[CL93] B. Cabral and L. C. Leedom. Imaging vector fields using line integral convolution. In *SIGGRAPH '93 : Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 263–270. ACM Press, 1993.

[CM96] J. Cupitt and K. Martinez. Vips : An image processing system for large images. In *Proceedings of SPIE*, 1996.

[DDG98a] F. Dupont, M. P. Deseilligny, and M. Gondran. Automatic interpretation of contour lines by using external data. In *WACV '98 : Proceedings of the 4th*

IEEE Workshop on Applications of Computer Vision (WACV'98), page 200. IEEE Computer Society, 1998.

[DDG98b] F. Dupont, M. P. Deseilligny, and M. Gondran. Automatic interpretation of scanned maps : Reconstruction of contour lines. In *GREC '97 : Selected Papers from the Second International Workshop on Graphics Recognition, Algorithms and Systems*, pages 194–206. Springer-Verlag, 1998.

[DDG99] F. Dupont, M. P. Deseilligny, and M. Gondran. Dtm extraction from topographic maps. In *ICDAR '99 : Proceedings of the Fifth International Conference on Document Analysis and Recognition*, page 475. IEEE Computer Society, 1999.

[DGE04] J. Diepstraten, M. Gorke, and T. Ertl. Remote line rendering for mobile devices. In *Computer Graphics International 2004 (CGI'04)*, pages 454–461, 2004.

[DGF03] S. Dusan, G. Gadbois, and J. Flanagan. Multimodal interaction on pda's integrating speech and pen inputs. In *Proceedings of EUROSPEECH*, pages 2225–2228, 2003.

[DMS+02] G. Dreyfus, J.-M. Martinez, M. Samuelides, M. B. Gordon, F. Badran, S. Thiria, and L. Hérault. *Réseaux de neurones - Méthodologie et applications*. Eyrolles, 2002.

[DRM99] T. K. Dey, E. A. Ramos, and K. Mehlhorn. Curve reconstruction : connecting dots with good reason. In *Proceedings of the 15th Symposium on Computational Geometry*, pages 197–206, 1999.

[Duc77] J. Duchon. Splines minimizing rotation-invariant semi-norms in sobolev spaces. In *Constructive Theory of Functions of Several Variables*. Springer-Verlag, 1977.

[Dup99] F. Dupont. *Contribution à l'analyse automatique de documents géographiques scannés : extraction de l'altimétrie*. PhD thesis, Thèse de doctorat de l'Université Paris 9, 1999.

[Dut84] G. Dutton. Geodesic modelling of planetary relief. *Cartographica*, 21 :188–207, 1984.

[DW00] T. K. Dey and R. Wenger. Reconstruction curves with sharp corners. In *SCG '00 : Proceedings of the sixteenth annual symposium on Computational geometry*, pages 233–241, 2000.

[DWS+97] M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich, and M. B. Mineev-Weinstein. ROAMing terrain : Real-time Optimally Adapting Meshes. In *Proceedings of the conference on Visualization '97*, pages 81–88. ACM Press, 1997.

[EAK95] L. Eikvil, K. Aas, and H. Koren. Tools for interactive map conversion and vectorization. *Third International Conference on Document Analysis and Recognition : ICDAR 1995*, 02 :927, 1995.

[EBSC99] L. Encarnação, O. Bimber, D. Schmalstieg, and S. Chandler. A translucent sketchpad for the virtual table exploring motion-based gesture recognition. In *Proceedings of Eurograhics '99*, pages 179–190, 1999.

[Edm65] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17 :449–667, 1965.

[ESE00] K. Engel, O. Sommer, and T. Ertl. A Framework for Interactive Hardware Accelerated Remote 3D-Visualization. In *Proceedings of EG/IEEE TCVG Symposium on Visualization VisSym '00*, pages 167–177,291, 2000.

[ESEE99] K. Engel, O. Sommer, C. Ernst, and T. Ertl. Remote 3D Visualization using Image-Streaming Techniques. In *Advances in Intelligent Computing and Multimedia Systems (ISIMADE '99)*, pages 91–96, 1999.

[FH05] M. S. Floater and K. Hormann. Surface parameterization : a tutorial and survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 157–186. Springer, 2005.

[Fit54] P. M. Fitts. The information capacity of the human motor system in controlling aplitude of movement. *Journal of Experimental Psychology*, 47 :386–392, 1954.

[Fit93] G. W. Fitzmaurice. Situated information spaces and spatially aware palmtop computers. *Communications of the ACM*, 1993.

[FJJ03] J. Ferraiolo, F. Jun, and D. Jackson. Scalable Vector Graphics (SVG) 1.1 Specification, 2003.

[FLW04] D. Fallman, A. Lund, and M. Wiberg. Scrollpad : Tangible scrolling with mobile devices. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04)*, 2004.

[FZC93] G. W. Fitzmaurice, S. Zhai, and M. H. Chignell. Virtual reality for palmtop computers. *ACM Trans. Inf. Syst.*, 11(3) :197–218, 1993.

[Gab74] H. N. Gabow. *Implementation of algorithms for maximum matching on nonbipartite graphs.* PhD thesis, Stanford University, 1974.

[GB05] T. Grossman and R. Balakrishnan. The bubble cursor : enhancing target acquisition by dynamic resizing of the cursor's activation area. In *CHI '05 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–290, 2005.

[GBBL04] Y. Guiard, R. Blanch, and M. Beaudouin-Lafon. Object pointing : a complement to bitmap pointing in GUIs. In *GI '04 : Proceedings of the 2004 conference on Graphics Interface*, pages 9–16, 2004.

[GF98] M. Gousie and W. Franklin. Converting elevation contours to a grid. *Eighth International Symposium on Spatial Data Handling (SDH)*, 1998.

[GF03] M. B. Gousie and W. R. Franklin. Constructing a DEM from grid-based data by computing intermediate contours. In *Proceedings of the eleventh ACM international symposium on Advances in geographic information systems*, pages 71–77, 2003.

[GR05] D. Gil and P. Radeva. Extending anisotropic operators to recover smooth shapes. *Computer Vision and Image Understanding*, 99(1) :110–125, 2005.

[Gre87] D. Greenlee. Raster and vector processing for scanned line work. *Photogrammetric Engineering and Remote Sensing*, 53(10) :1383–1387, 1987.

[GT04] J. Gong and P. Tarasewich. Guidelines for handheld mobile device interface design. In *Proceedings of DSI 2004*, 2004.

[GW92] R. Gonzalez and R. Woods. *Digital Image Processing*, pages 518–548. Addison-Wesley Publishing Company, 1992.

[Har71] R. L. Hardy. Multiquadric equations of topography and other irregular surfaces. *J. Geophys. res*, 76 :1905–1915, 1971.

[HN89] R. Halstead-Nussloch. The design of phone-based interfaces for consumers. In *CHI '89 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 374–352, 1989.

[Hop96] H. Hoppe. Progressive meshes. *Proceedings of SIGGRAPH 96*, pages 99–108, 1996.

[Hop98] H. Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. In *IEEE Visualization '98*, pages 35–42, 1998.

[HPG05] M. Hachet, J. Pouderoux, and P. Guitton. A camera-based interface for interaction with mobile handheld computers. In *Proceedings of I3D'05 - ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games*, pages 65–71, 2005.

[HPGG05] M. Hachet, J. Pouderoux, P. Guitton, and J.-C. Gonzato. Tangimap - a tangible interface for visualization of large documents on handheld computers. In *Proceedings of Graphics Interface*, pages 9–15, 2005.

[HPKG07] M. Hachet, J. Pouderoux, S. Knödel, and P. Guitton. 3d panorama service on mobile device for hiking. In *Proceedings of CHI 2007 Workshop on Mobile Spatial Interaction*, 2007.

[HPSH00] K. Hinckley, J. S. Pierce, M. Sinclair, and E. Horvitz. Sensing techniques for mobile interaction. In *Proceedings of the 13th annual ACM symposium on User Interface Software and Technology (UIST 2000)*, pages 91–100, 2000.

[HPTG07] M. Hachet, J. Pouderoux, F. Tyndiuk, and P. Guitton. 'jump and refine' for rapid pointing on mobile phones. In *CHI 2007 Notes Conference Proceedings*, 2007.

[HSS03] K. Hormann, S. Spinello, and P. Schröder. C^1-continuous terrain reconstruction from sparse contours. In *Proceedings of Vision, Modeling, and Visualization 2003*, pages 289–297, 2003.

[Hut88] M. Hutchinson. Calculation of hydrologically sound digital elevation models. In I. G. Union, editor, *Proceedings of the Third International Symposium on Spatial Data Handling*, pages 117–133, 1988.

[IH00] T. Igarashi and K. Hinckley. Speed-dependent automatic zooming for browsing large documents. In *Proceedings of the 13th annual ACM symposium on User Interface Software and Technology (UIST 2000)*, pages 139–148. ACM Press, 2000.

[JH78] A. G. Journel and C. J. Huijbregts. *Mining Geostatistics*. Academic Press, 1978.

[JHJ86] T. A. Jones, D. Hamilton, and C. Johnson. *Contouring geologic surfaces with the computer*. Van Norstand Reinhold Company Inc., 1986.

[JL97] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. In *Proceedings of the 8^{th} Eurographics Workshop on Visualization in Scientific Computing*, pages 45–55, 1997.

[KB95] P. Kabbash and W. A. S. Buxton. The "prince" technique : Fitts' law and selection using area cursors. In *CHI '95 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 273–279, 1995.

[KBS05] A. K. Karlson, B. B. Bederson, and J. SanGiovanni. AppLens and launchTile : two designs for one-handed thumb use on small devices. In *CHI '05 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 201–210. ACM Press, 2005.

[KI98] K. Kawachiya and H. Ishikawa. NaviPoint : an input device for mobile information browsing. In *CHI '98 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1–8. ACM Press/Addison-Wesley Publishing Co., 1998.

[KMRH03] D. M. Krum, R. Melby, W. Ribarsky, and L. Hodges. Isometric pointer interfaces for wearable 3D visualization. In *CHI '03 extended abstracts on Human factors in computing systems*, pages 774–775. ACM Press, 2003.

[Koh89] T. Kohonen. *Self-Organization and associative memory.* Springer-Verlag, 3rd edition, 1989.

[Kri51] D. G. Krige. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Chem., Metal. and Mining Soc. of South Africa*, 52(6) :119–139, 1951.

[KS02] J. Kjedskov and M. B. Skov. Interaction design for handheld computers. In *Proceedings of the 5th Asian Pacific Conference on Human-Computer Interaction, APCHI 2002*. Science Press, 2002.

[KZ96] A. Khotanzad and E. Zink. Color paper map segmentation using eigenvector line-fitting. In *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 190–194, 1996.

[KZ03] A. Khotanzad and E. Zink. Contour line and geographic feature extraction from usgs color topographical paper maps. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(1) :18–31, 2003.

[LC03] B. S. Larsen and N. J. Christensen. Real-time terrain rendering using smooth hardware optimized level of detail. In *WSCG*, 2003.

[Lev02] J. Levenberg. Fast view-dependent level-of-detail rendering using cached geometry. In *VIS '02 : Proceedings of the conference on Visualization '02*, pages 259–266, 2002.

[LGMR99] P. A. Longley, M. F. Goodchild, D. J. Maguire, and D. W. Rhind, editors. *Geographical Information Systems : Principles, Techniques, Management and Applications.* John Wiley, 2nd edition, 1999.

[LH04] F. Losasso and H. Hoppe. Geometry clipmaps : terrain rendering using nested regular grids. *ACM Trans. Graph.*, 23(3) :769–776, 2004.

[LKR+96] P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hughes, N. Faust, and G. Turner. Real-time, continuous level of detail rendering of height fields. In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 109–118. ACM SIGGRAPH / Addison Wesley, 1996.

[LM84] C. Lantuéjoul and F. Maisonneuve. Geodesic methods in image analysis. *Pattern Recognition*, 17 :177–187, 1984.

[LO82] F. Leberl and D. Olson. Raster scanning for operational digitizing of graphical data. *Photogrammetric Engineering and Remote Sensing*, 48(4) :615–627, 1982.

[LP01] P. Lindstrom and V. Pascucci. Visualization of large terrains made easy. In *VIS '01 : Proceedings of the conference on Visualization '01*, pages 363–371. IEEE Computer Society, 2001.

[LVdV05] J.-O. Lachaud, A. Vialard, and F. de Vieilleville. Analysis and comparative evaluation of discrete tangent estimators. In *Proceedings of Int. Conf. Discrete Geometry for Computer Imagery (DGCI'2005)*, pages 140–251, 2005.

[LWC+02] D. Luebke, B. Watson, J. D. Cohen, M. Reddy, and A. Varshney. *Level of Detail for 3D Graphics*. Elsevier Science Inc., 2002.

[Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Symposium on Math, Statistics, and Probability*, volume 1, pages 281–297, 1967.

[Mac92] I. S. MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7 :91–139, 1992.

[Mar04] J.-E. Marvie. *Visualisation Interactive d'Environnements Virtuels Complexes à travers des Réseaux et sur des Machines à Performances Variables*. PhD thesis, INSA de Rennes, France, 2004.

[Mat71] G. Matheron. La théorie des variables généralisées et ses applications. *Les Cahiers du Centre de Morphologie Mathématique de Fontainebleau*, 5, 1971.

[MB03] J.-E. Marvie and K. Bouatouch. Remote rendering of massively textured 3D scenes through progressive texture maps. In *The 3rd IASTED conference on Visualisation, Imaging and Image Processing*, volume 2, pages 756–761, 2003.

[Min98] M. Mine. *Exploiting proprioception in virtual-environment interaction*. PhD thesis, University of North Carolina at Chapel Hill, 1998.

[Paj98] R. Pajarola. Large scale terrain visualization using the restricted quadtree triangulation. In *Proceedings IEEE Visualization'98*, pages 19–26, 1998.

[PC75] T. Peucker and N. Chrisman. Cartographic data structures. *The American Cartographer*, 2 :55–69, 1975.

[PD94] M. Pierrot-Deseilligny. *Lecture automatique de cartes*. PhD thesis, Thèse de doctorat de l'Université Paris 5, 1994.

[PDSS98] M. Pierrot-Deseilligny, G. Stamon, and C. Y. Suen. Veinerization : A new shape description for flexible skeletonization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5) :505–521, 1998.

[PFTV92] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, 1992.

[PGGG04] J. Pouderoux, J.-C. Gonzato, P. Guitton, and X. Granier. A software for reconstructing 3d-terrains from scanned maps, 2004. ACM SIGGRAPH 2004 Sketches and Applications.

[PGPG07] J. Pouderoux, J.-C. Gonzato, A. Pereira, and P. Guitton. Toponym recognition in scanned color topographic maps. In *Proceedings of ICDAR 2007 : 9th International Conference on Document Analysis and Recognition*, pages 531–535, 2007.

[PM05] J. Pouderoux and J.-E. Marvie. Adaptive streaming and rendering of large terrains using strip masks. In *Proceedings of ACM GRAPHITE 2005*, pages 299–306, 2005.

[PMA03] J. Pierce, H. Mahaney, and G. Abowd. Opportunistic annexing for handheld devices : Opportunities and challenges. Technical report, Georgia Institute of Technology GIT-GVU-03-31, 2003.

[PS82] C. C. Paige and M. A. Saunders. Lsqr : An algorithm for sparse linear equations and sparse least squares. *TOMS*, 8(1) :43–71, 1982.

[PS07] J. Pouderoux and S. Spinello. Global contour lines reconstruction in topographic maps. In *Proceedings of ICDAR 2007 : 9th International Conference on Document Analysis and Recognition*, pages 779–783, 2007.

[PTGG04] J. Pouderoux, I. Tobor, J.-C. Gonzato, and P. Guitton. Adaptive hierarchical RBF interpolation for creating smooth digital elevation models. In *Proceedings of the Twelfth ACM International Symposium on Advances in Geographical Information System 2004*, pages 232–240. ACM Press, 2004.

[Pto89] C. Ptolémée. *Traité de géographie - traduction de l'Abbé Halma.* Albert Blanchard, 1989.

[RDE99] S. Riley, S. DeGloria, and R. Elliot. A terrain ruggedness index that quantifies topographic heterogeneity. *Intermountain Journal of Sciences*, 5 :23–27, 1999.

[Rek96] J. Rekimoto. Tilting operations for small screen interfaces. In *ACM Symposium on User Interface Software and Technology (UIST 96)*, pages 167–168, 1996.

[RHSS98] S. Roettger, W. Heidrich, P. Slusallek, and H.-P. Seidel. Real-Time Generation of Continuous Levels of Detail for Height Fields. In *Procceedings of WSCG '98*, pages 315–322, 1998.

[RIL00] M. Reddy, L. Iverson, and Y. G. Leclerc. Under the hood of geovrml 1.0. In *VRML '00 : Proceedings of the fifth symposium on Virtual reality modeling language (Web3D-VRML)*, pages 23–28. ACM Press, 2000.

[RLIB99] M. Reddy, Y. Leclerc, L. Iverson, and N. Bletter. TerraVision II : Visualizing massive terrain databases in VRML. *IEEE Computer Graphics and Applications*, 19(2) :30–38, 1999.

[RM00] X. Ren and S. Moriya. Improving selection performance on pen-based systems : a study of pen-based interaction for selection tasks. *ACM Transaction on Computer-Human Interaction*, 7(3) :384–416, 2000.

[RN95] J. Rekimoto and K. Nagao. The world through the computer : Computer augmented interaction with real world environments. In *ACM Symposium on User Interface Software and Technology (UIST 1995)*, pages 29–36, 1995.

[Roh04] M. Rohs. Real-world interaction with camera-phones. In *Proceedings of the 2nd International Symposium on Ubiquitous Computing Systems (UCS 2004)*, 2004.

[Ros58] F. Rosenblatt. The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6) :386–408, 1958.

[RP03] T. Rantakokko and J. Plomp. An adaptive map-based interface for situated services. In *Proceedings of Smart Object Conference*, 2003.

[Ser82] J. Serra. *Image analysis and mathematical morphology.* Academic Press, 1982.

[SG04] S. Spinello and P. Guitton. Contour line recognition from scanned topographic maps. In *Proceedings of Winter School of Computer Graphics : WSCG*, pages 419–426, 2004.

[She68] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pages 517–524. ACM Press, 1968.

[She96] J. R. Shewchuk. Triangle : Engineering a 2D quality mesh generator and delaunay triangulator. In *Applied Computational Geometry : Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, 1996.

[Shn97] B. Shneiderman. *Designing the User Interface : Strategies for Effective Human-Computer Interaction*. Addison-Wesley Longman Publishing Co., Inc., 1997.

[Sib80] R. Sibson. A vector identity for the dirichlet tesselation. *Math. Proc. Camb. Phil. Soc.*, 87 :151–155, 1980.

[Sib81] R. Sibson. *A Brief Description of Natural Neighbor Interpolation*, pages 21–36. V. Barnett, John Wiley and Sons, 1981.

[SMK01] M. Silfverberg, I. S. MacKenzie, and T. Kauppinen. An isometric joystick as a pointing device for handheld information terminals. In *GI'01 : Proccedings of Graphics Interface 2001*, pages 119–126. Canadian Information Processing Society, 2001.

[Soi91] P. Soille. Spatial distributions from contour lines : An efficient methodology based on distance transformation. *Journal of Visual Communication and Image Representation*, 2(2) :138–150, 1991.

[Soi92] P. Soille. *Morphologie mathématique : du relief à la dimensionalité - algorithmes et méthodes*. PhD thesis, Université catholique de Louvain et Ecole nationale supérieure des Mines de Paris, 1992.

[SS02] T. Strothotte and S. Schlechtweg. *Non-photorealistic computer graphics : modeling, rendering, and animation*. Morgan Kaufmann Publishers Inc., 2002.

[TG00] D. Thibault and C. M. Gold. Terrain reconstruction from contours by skeleton construction. *Geoinformatica*, 4(4) :349–373, 2000.

[TMJ98] C. C. Tanner, C. J. Migdal, and M. T. Jones. The clipmap : a virtual mipmap. In *SIGGRAPH '98 : Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 151–158, 1998.

[TRS04] I. Tobor, P. Reuter, and C. Schlick. Efficient reconstruction of large scattered geometric datasets using the partition of unity and radial basis functions. In *WSCG (Winter School of Computer Graphics)*, 2004.

[Vel02] A. Velazquez. *Location, Recuperation and Identification of Alphanumeric Character Layer into Color Cartographic Maps*. PhD thesis, National Polytechnic Institute of Mexico, 2002.

[Vin91] L. Vincent. Efficient computation of various types of skeletons. In *Medical Imaging V : Image Processing*, volume SPIE-1445, pages 297–311, 1991.

[Wat92] D. F. Watson. *Contouring : A Guide to the Analysis and Display of Spatial Data*. Pergammon Press, 1992.

[WMB03] E. Woods, P. Mason, and M. Billinghurst. Magicmouse : an inexpensive 6-degree-of-freedom mouse. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Austalasia and South East Asia*, pages 285–286. ACM Press, 2003.

[Woo96] J. D. Wood. *The Geomorphological Characterisation of Digital Elevation Models*. PhD thesis, University of Leicester, UK, 1996.

[WS03] D. Wagner and D. Schmalstieg. First steps towards handheld augmented reality. In *Proceedings of Seventh IEEE International Symposium on Wearable Computers*, 2003.

[Xia89] I. Xia. Skeletonization via the realisation of the fire front's propagation and extintion in digital binary shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11 :1076–1086, 1989.

[Yan93] H. Yan. Color map image segmentation using optimized nearest neighbor classifiers. In *Proceedings of ICDAR'93 : International Conference on Document Analysis and Recognition*, pages 111–114, 1993.

[Yee03] K.-P. Yee. Peephole displays : pen interaction on spatially aware handheld computers. In *Proceedings of the conference on Human factors in computing systems*, pages 1–8. ACM Press, 2003.

[ZZSP01] Y. Zhao, J. Zhou, J. Shi, and Z. Pan. A fast algorithm for large scale terrain walkthrough. In *Proceedings of International Conference on CAD and Graphics 2001*, 2001.

Webographie

[AutoDEM] Site officiel d'AutoDEM.
http://glutes.sourceforge.net.

[ArcGIS] Site officiel de la plate-forme ArcGIS d'ESRI.
http://www.esri.com/software/arcgis/.

[GDAL] GDAL : Geospatial Data Abstraction Library.
http://www.gdal.org.

[GLUTES] Site officiel de GLUT—ES.
http://glutes.sourceforge.net.

[GRASS] Site officiel de GRASS GIS.
http://grass.itc.it.

[LGMA] Large Geometric Models Archive at Georgia Tech.
http://www-static.cc.gatech.edu/projects/large_models.

[MapServer] Site officiel de MapServer.
http://mapserver.gis.umn.edu.

[MobiTouch] Site officiel de MobiTouch.
http://www.synaptics.com/products/mobto.cfm.

[OGC] Site officiel d'OGC : Open Geospatial Consortium.
http://www.opengeospatial.org.

Publications

[PGG03] **J. Pouderoux**, J.-C. Gonzato, P. Guitton. Création semi-automatique d'un Modèle Numérique de Terrain, 2003. In *Actes des 16ièmes Journées de l'Association Française d'Informatique Graphique (AFIG)*, pages 151–160.

[PGGG04] **J. Pouderoux**, J.-C. Gonzato, P. Guitton, and X. Granier. A software for reconstructing 3d-terrains from scanned maps, 2004. ACM SIGGRAPH 2004 Sketches and Applications.

[PTGG04] **J. Pouderoux**, I. Tobor, J.-C. Gonzato, and P. Guitton. Adaptive hierarchical RBF interpolation for creating smooth digital elevation models. In *Proceedings of the Twelfth ACM International Symposium on Advances in Geographical Information System 2004*, pages 232–240. ACM Press, 2004.

[PM05] **J. Pouderoux** and J.-E. Marvie. Adaptive streaming and rendering of large terrains using strip masks. In *Proceedings of ACM GRAPHITE 2005*, pages 299–306, 2005.

[HPG05] M. Hachet, **J. Pouderoux**, and P. Guitton. A camera-based interface for interaction with mobile handheld computers. In *Proceedings of I3D'05 - ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games*, pages 65–71, 2005.

[HPGG05] M. Hachet, **J. Pouderoux**, P. Guitton, and J.-C. Gonzato. Tangimap - a tangible interface for visualization of large documents on handheld computers. In *Proceedings of Graphics Interface*, pages 9–15, 2005.

[HPTG07] M. Hachet, **J. Pouderoux**, F. Tyndiuk, and P. Guitton. 'Jump and Refine' for rapid pointing on mobile phones. In *CHI 2007 Notes Conference Proceedings*, 2007.

[HPKG07] M. Hachet, **J. Pouderoux**, S. Knödel, and P. Guitton. 3D panorama service on mobile device for hiking. In *Proceedings of CHI 2007 Workshop on Mobile Spatial Interaction*, pages 101–104, 2007.

[PGPG07] **J. Pouderoux**, J.-C. Gonzato, A. Pereira, and P. Guitton. Toponym recognition in scanned color topographic maps. In *Proceedings of ICDAR 2007 : 9th International Conference on Document Analysis and Recognition*, pages 531–535, 2007.

[PS07] **J. Pouderoux** and S. Spinello. Global contour lines reconstruction in topographic maps. In *Proceedings of ICDAR 2007 : 9th International Conference on Document Analysis and Recognition*, pages 779–783, 2007.